

PROYECTO PROGRAMACIÓN DE REDES

Acosta Romero Martin Gustavo y Hernández Durán Osmar

ingeniería
informática

Contenido

Introducción	3
Objetivos y alcance.....	3
Objetivos.....	3
Alcance	4
Requisitos	5
Requisitos funcionales	5
Requisitos no funcionales	6
Diseño	7
Arquitectura utilizada	7
Explicación de la elección de tecnologías.....	7
Decisiones de implementación	8
Diagramas UML.....	9
Implementación.....	11
Dependencias.....	11
Plugins	12
Java beans	13
Alumnos.....	13
Grupos	13
Grupo_Alumnos	15
Maestros	16
Mensajes	17
DAO (Data Access Object).....	18
Estructura básica de un DAO.....	18
RMI	30
Servlets	33
Función de los Servlets	33
Servlets ocupados en el proyecto	33
Resultados.....	43
Inicio de sesión.....	43
Registro.....	43

Menú para alumnos	44
Menú para maestros	44
Perfil	45
Chat de alumnos.....	45
Chat de maestros	46
Agregar alumnos a un grupo (Maestro)	46
Ver alumnos de un grupo (Maestro)	47
Instrucciones.....	48
Inicio de sesión	48
Registro.....	48
Menú para alumnos	49
Menú para maestros	49
Perfil.....	50
Chat de alumnos.....	50
Chat de maestros	51
Agregar alumnos a un grupo (Maestro)	51
Ver alumnos de un grupo (Maestro)	52
Conclusiones	53
Referencias	54

Introducción

En la actualidad, la necesidad de herramientas digitales para la comunicación efectiva y la colaboración en entornos educativos ha crecido exponencialmente. Aplicaciones como MS Teams y WhatsApp han demostrado ser esenciales en la interacción entre docentes y estudiantes. Inspirándonos en estas herramientas, nuestro proyecto busca crear una aplicación web que combine funcionalidad de canales de WhatsApp y facilidad de uso para satisfacer las necesidades del entorno educativo.

La aplicación se centra en ofrecer una plataforma de comunicación controlada, en la que los maestros tienen el rol principal en los chats, pudiendo enviar mensajes y compartir fotos con sus estudiantes. Esta manera de comunicación unidireccional garantiza el orden y la claridad al momento de transmitir la información.

Objetivos y alcance

Objetivos

1. Diseñar e implementar una aplicación web que facilite la comunicación entre maestros y estudiantes mediante canales controlados.
2. Integrar protocolos de comunicación como RMI y sockets para garantizar un rendimiento eficiente y escalable.
3. Implementar una interfaz sencilla y accesible que permita a los usuarios interactuar con la aplicación de manera intuitiva.
4. Asegurar que la plataforma permita el envío de mensajes y fotos.

Alcance

Este proyecto está diseñado principalmente para entornos educativos donde se requiere una comunicación clara. La aplicación incluye las siguientes funcionalidades:

1. Canales de chat donde solo los maestros pueden publicar mensajes y multimedia.
2. Seguridad en las conexiones y transferencia de datos mediante el uso de protocolos seguros, aprovechando las capacidades de RMI (Remote Method Invocation) para garantizar comunicaciones seguras entre los componentes distribuidos del sistema.
3. Escalabilidad para soportar un número significativo de usuarios concurrentes.

La aplicación no busca sustituir plataformas de comunicación masiva, realmente busca ofrecer una solución específica y eficiente para entornos académicos con cierto desorden en su comunicación sin arriesgarse a ocupar contactos personales.

Requisitos

Requisitos funcionales

Los maestros y alumnos pueden ver sus datos personales en el apartado de perfil.

La aplicación contiene un sistema de inicio de sesión (login) que solicita la matrícula o número de control (dependiendo si es maestro o alumno), además de la contraseña.

Incluye un apartado de registro que solicita los datos personales de los usuarios.

Solo los maestros pueden:

1. Crear grupos de chat.
2. Agregar alumnos al grupo.
3. Eliminar alumnos del grupo.
4. Ver quiénes están en el grupo.
5. Eliminar el grupo o modificar su nombre.
6. Enviar mensajes y fotos en los grupos.

Los alumnos pueden:

1. Consultar sus datos personales en el apartado de perfil.
2. Acceder a los chats para ver los mensajes enviados en los grupos a los que pertenecen, pero no pueden enviar mensajes.
3. Todas las operaciones relacionadas con el acceso a la base de datos se realizan mediante RMI para asegurar una comunicación eficiente y segura.

La mensajería en tiempo real utiliza WebSockets para garantizar una experiencia fluida y actualizaciones instantáneas, los navegadores no soportan sockets TCP ni UDP.

Requisitos no funcionales

1. Escalabilidad: La arquitectura del sistema debe permitir el crecimiento, tanto en número de usuarios como en funcionalidad.
2. Compatibilidad: La aplicación debe ser accesible desde los principales navegadores web y dispositivos.
3. Mantenibilidad: El código fuente debe ser modular y documentado para facilitar futuras modificaciones y expansiones.
4. Usabilidad: La interfaz de usuario debe ser intuitiva y fácil de usar, con diseño responsive para adaptarse a diferentes tamaños de pantalla.
5. Confiabilidad: El sistema debe garantizar la integridad de los datos y minimizar la posibilidad de errores en las operaciones realizadas.

Diseño

Arquitectura utilizada

El sistema está basado en una arquitectura cliente-servidor, utilizando los siguientes componentes:

1. Controladores Servlets: Gestionan las solicitudes del cliente y las respuestas del servidor.
2. Clases DAO (Data Access Object): Se encargan de interactuar con la base de datos para realizar operaciones CRUD.
3. Java Beans: Representan las entidades del sistema, facilitando la transferencia de datos entre las capas.
4. Interfaces RMI: Implementan el servidor RMI para proporcionar acceso remoto seguro a las operaciones de la base de datos.

Explicación de la elección de tecnologías

Las tecnologías utilizadas incluyen Tomcat, RMI, WebSockets, Java, JavaScript, HTML y CSS. Estas fueron seleccionadas debido a que son las herramientas principales aprendidas durante la carrera y permiten desarrollar una aplicación robusta, escalable y compatible con diferentes plataformas.

1. Tomcat: Facilita el despliegue de aplicaciones web basadas en Java.
2. RMI: Proporciona un mecanismo confiable para la comunicación remota entre el cliente y el servidor.
3. WebSockets: Garantizan una mensajería en tiempo real eficiente y fluida.
4. Java, JavaScript, HTML, CSS: Forman la base para la lógica del negocio, interacción del cliente y diseño de la interfaz de usuario.

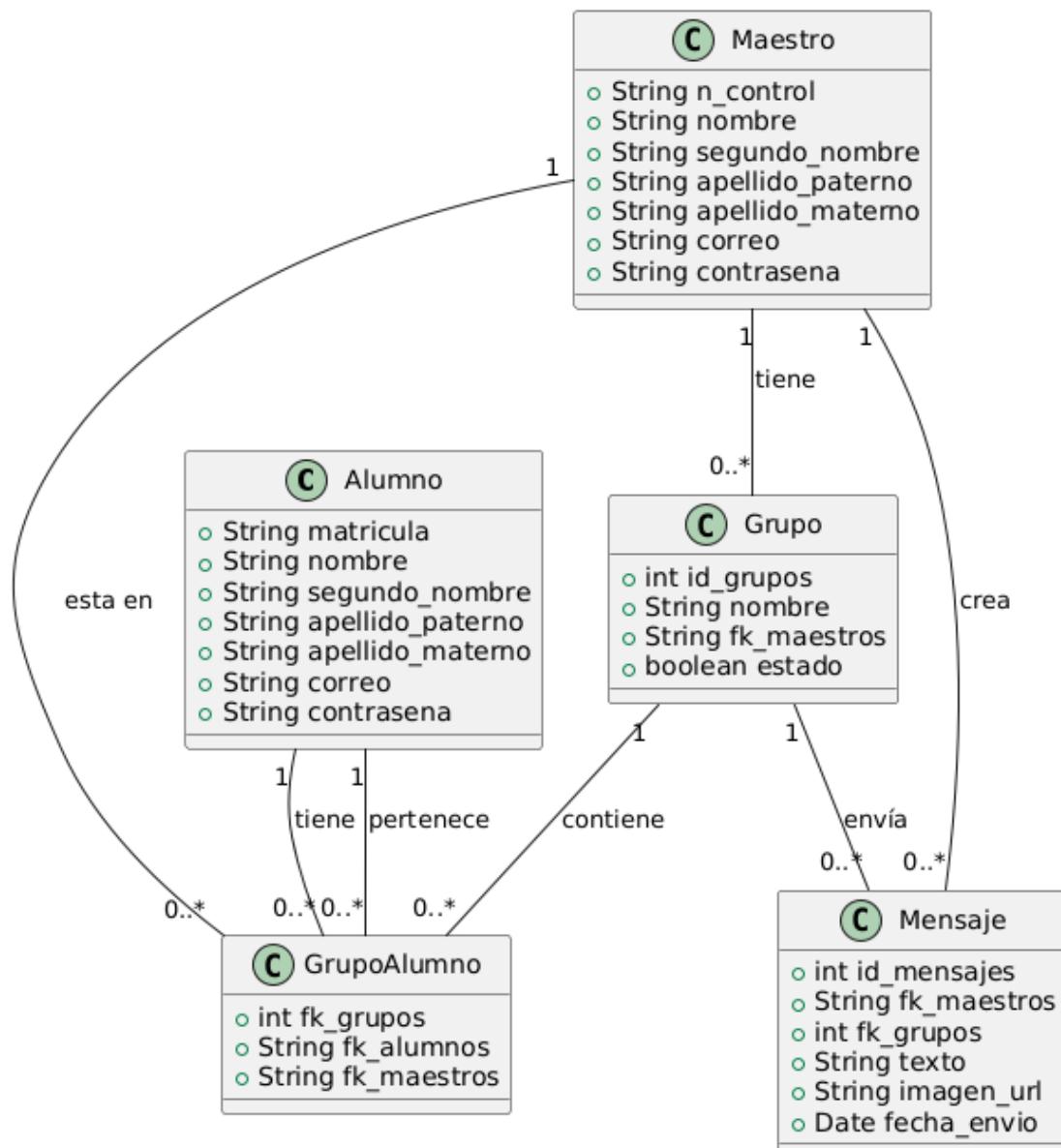
Decisiones de implementación

1. División clara de las capas de la aplicación (presentación, lógica de negocio y acceso a datos) para mejorar la mantenibilidad.
2. Uso de RMI para garantizar la seguridad y eficiencia en la comunicación con la base de datos.
3. Implementación de WebSockets para una mensajería en tiempo real.
4. Estructuración modular del código para facilitar la extensión y la depuración.

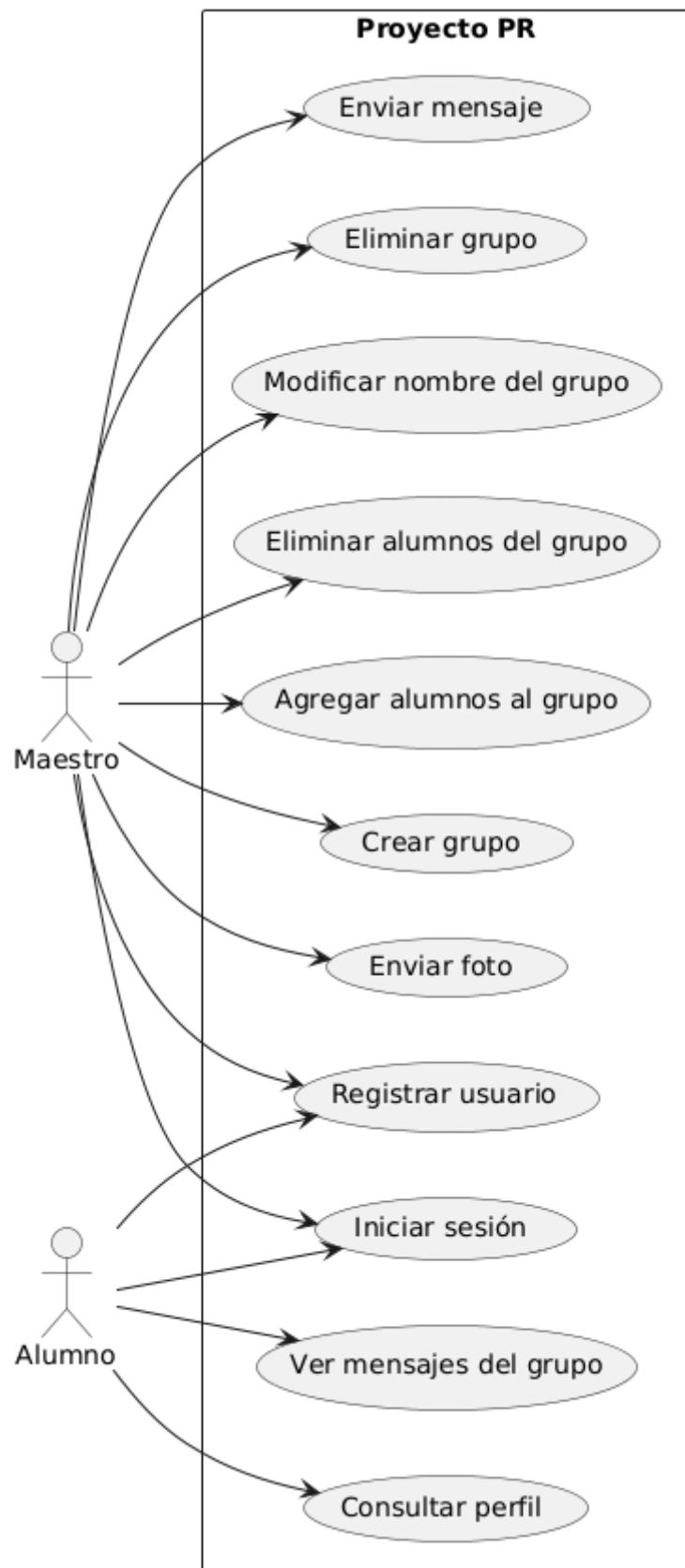
Diagramas UML

Para representar el diseño del sistema, se incluyen los siguientes diagramas UML:

- Diagrama de clases:



- Diagrama de casos de uso:



Implementación

Dependencias

1. **JUnit** es una biblioteca utilizada para realizar pruebas unitarias en Java. En este caso, se usa en el ámbito de pruebas (scope = test), lo que significa que solo se incluirá en el classpath cuando se ejecuten pruebas.
2. **mysql-connector-j** es un controlador JDBC para conectar tu aplicación Java con una base de datos MySQL. Esto te permite ejecutar consultas SQL y realizar operaciones CRUD en la base de datos.
3. **javax.servlet-api** es una API que permite crear y gestionar servlets. Los servlets son clases Java que responden a solicitudes web, procesando datos de formularios y generando respuestas. La versión 4.0.1 es compatible con las aplicaciones web basadas en el estándar Servlet 4.0. El alcance provided indica que este JAR será proporcionado por el servidor de aplicaciones (por ejemplo, Tomcat).
4. **javaee-web-api** es una API de Java EE que proporciona clases para crear aplicaciones web en un entorno de servidor. Al igual que la dependencia anterior, tiene un alcance provided, lo que significa que se espera que esta biblioteca sea proporcionada por el servidor de aplicaciones.
5. **Commons DBCP2** es una biblioteca de Apache que proporciona un conjunto de utilidades para el manejo de conexiones a bases de datos. Esto es útil para gestionar conexiones en un pool, lo que mejora el rendimiento y la escalabilidad al evitar la sobrecarga de crear nuevas conexiones cada vez que se interactúa con la base de datos.

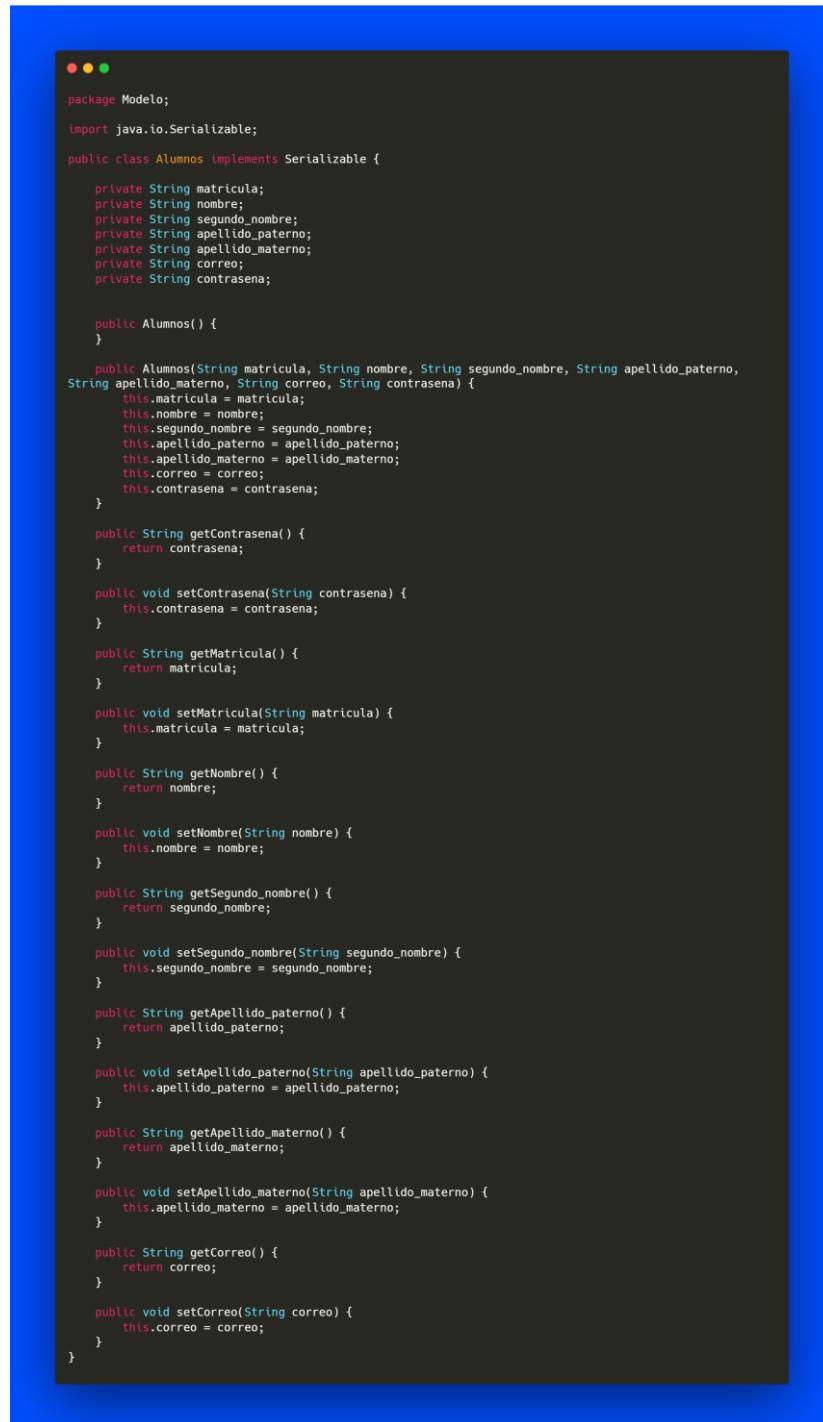
Plugins

1. **tomcat7-maven-plugin** se utiliza para desplegar la aplicación directamente en un servidor Tomcat desde Maven. Esto permite realizar el despliegue de la aplicación web sin necesidad de mover archivos manualmente al servidor. En el ejemplo, no se está configurando un puerto explícito, pero se podría personalizar si es necesario.
2. **maven-compiler-plugin** es responsable de compilar el código fuente de tu proyecto. En este caso, se está configurando para usar la versión de Java 11 tanto para la compilación (source) como para la ejecución (target).
3. **maven-war-plugin** se utiliza para empaquetar tu aplicación web en un archivo WAR (Web ARchive). Este archivo WAR contiene todos los archivos necesarios para desplegar la aplicación en un servidor web. Este plugin configura el empaquetado para que el resultado sea un archivo .war.

Java beans

Son las representaciones de las Tablas en la base de datos, con sus respectivos métodos getters y setters, haciendo uso de los conocimientos adquiridos en Paradigmas de programación.

Alumnos



```
● ● ●
package Modelo;

import java.io.Serializable;

public class Alumnos implements Serializable {

    private String matricula;
    private String nombre;
    private String segundo_nombre;
    private String apellido_paterno;
    private String apellido_materno;
    private String correo;
    private String contraseña;

    public Alumnos() {
    }

    public Alumnos(String matricula, String nombre, String segundo_nombre, String apellido_paterno,
String apellido_materno, String correo, String contraseña) {
        this.matricula = matricula;
        this.nombre = nombre;
        this.segundo_nombre = segundo_nombre;
        this.apellido_paterno = apellido_paterno;
        this.apellido_materno = apellido_materno;
        this.correo = correo;
        this.contraseña = contraseña;
    }

    public String getContraseña() {
        return contraseña;
    }

    public void setContraseña(String contraseña) {
        this.contraseña = contraseña;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getSegundo_nombre() {
        return segundo_nombre;
    }

    public void setSegundo_nombre(String segundo_nombre) {
        this.segundo_nombre = segundo_nombre;
    }

    public String getApellido_paterno() {
        return apellido_paterno;
    }

    public void setApellido_paterno(String apellido_paterno) {
        this.apellido_paterno = apellido_paterno;
    }

    public String getApellido_materno() {
        return apellido_materno;
    }

    public void setApellido_materno(String apellido_materno) {
        this.apellido_materno = apellido_materno;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }
}
```

Grupos

```
● ● ●

package Modelo;

import java.io.Serializable;

public class Grupos implements Serializable {
    private int id_grupos;
    private String nombre;
    private String fk_maestros;
    private boolean estado;

    public Grupos() {
    }

    public Grupos(int id_grupos, String nombre, boolean estado, String fk_maestros) {
        this.id_grupos = id_grupos;
        this.nombre = nombre;
        this.estado = estado;
        this.fk_maestros = fk_maestros;
    }

    public String getFk_maestros() {
        return fk_maestros;
    }

    public void setFk_maestros(String fk_maestros) {
        this.fk_maestros = fk_maestros;
    }

    public int getId_grupos() {
        return id_grupos;
    }

    public void setId_grupos(int id_grupos) {
        this.id_grupos = id_grupos;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public boolean isEstado() {
        return estado;
    }

    public void setEstado(boolean estado) {
        this.estado = estado;
    }
}
```

Grupo_Alumnos

```
● ● ●

package Modelo;

import java.io.Serializable;

public class Grupos_Alumnos implements Serializable {

    private int fk_grupos;
    private String fk_alumnos;
    private String fk_maestros;
    private String nombreMateria;

    public Grupos_Alumnos() {
    }

    public Grupos_Alumnos(int fk_grupos, String nombreMateria) {
        this.fk_grupos = fk_grupos;
        this.fk_alumnos = fk_alumnos;
        this.fk_maestros = fk_maestros;
        this.nombreMateria = nombreMateria;
    }

    public Grupos_Alumnos(int fk_grupos, String fk_alumnos, String fk_maestros) {
        this.fk_grupos = fk_grupos;
        this.fk_alumnos = fk_alumnos;
        this.fk_maestros = fk_maestros;
    }

    public Grupos_Alumnos(int fk_grupos, String fk_alumnos, String fk_maestros, String nombreMateria) {
        this.fk_grupos = fk_grupos;
        this.fk_alumnos = fk_alumnos;
        this.fk_maestros = fk_maestros;
        this.nombreMateria = nombreMateria;
    }

    public int getFk_grupos() {
        return fk_grupos;
    }

    public void setFk_grupos(int fk_grupos) {
        this.fk_grupos = fk_grupos;
    }

    public String getFk_alumnos() {
        return fk_alumnos;
    }

    public void setFk_alumnos(String fk_alumnos) {
        this.fk_alumnos = fk_alumnos;
    }

    public String getFk_maestros() {
        return fk_maestros;
    }

    public void setFk_maestros(String fk_maestros) {
        this.fk_maestros = fk_maestros;
    }

    public String getNombreMateria() {
        return nombreMateria;
    }

    public void setNombreMateria(String nombreMateria) {
        this.nombreMateria = nombreMateria;
    }
}
```

Maestros

```
● ● ●

package Modelo;

import java.io.Serializable;

public class Maestros implements Serializable {
    private String n_control;
    private String nombre;
    private String segundo_nombre;
    private String apellido_paterno;
    private String apellido_materno;
    private String correo;
    private String contrasena;

    public Maestros() {
    }

    public Maestros(String n_control, String nombre, String segundo_nombre, String apellido_paterno, String apellido_materno, String correo, String contrasena) {
        this.n_control = n_control;
        this.nombre = nombre;
        this.segundo_nombre = segundo_nombre;
        this.apellido_paterno = apellido_paterno;
        this.apellido_materno = apellido_materno;
        this.correo = correo;
        this.contrasena = contrasena;
    }

    public void setContrasena(String contrasena) {
        this.contrasena = contrasena;
    }

    public String getContrasena() {
        return contrasena;
    }

    public String getN_control() {
        return n_control;
    }

    public void setN_control(String n_control) {
        this.n_control = n_control;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getSegundo_nombre() {
        return segundo_nombre;
    }

    public void setSegundo_nombre(String segundo_nombre) {
        this.segundo_nombre = segundo_nombre;
    }

    public String getApellido_paterno() {
        return apellido_paterno;
    }

    public void setApellido_paterno(String apellido_paterno) {
        this.apellido_paterno = apellido_paterno;
    }

    public String getApellido_materno() {
        return apellido_materno;
    }

    public void setApellido_materno(String apellido_materno) {
        this.apellido_materno = apellido_materno;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }
}
```

Mensajes

```
● ● ●
package Modelo;

import java.io.Serializable;
import java.sql.Timestamp;

public class Mensajes implements Serializable {

    private int id_mensajes;
    private String fk_maestros;
    private int fk_grupos;
    private String texto;
    private String imagen_url;
    private Timestamp fecha_envio;

    public Mensajes() {
    }

    public Mensajes(int id_mensajes, String fk_maestros, int fk_grupos, String texto, String imagen_url, Timestamp fecha_envio) {
        this.id_mensajes = id_mensajes;
        this.fk_maestros = fk_maestros;
        this.fk_grupos = fk_grupos;
        this.texto = texto;
        this.imagen_url = imagen_url;
        this.fecha_envio = fecha_envio;
    }

    public int getId_mensajes() {
        return id_mensajes;
    }

    public void setId_mensajes(int id_mensajes) {
        this.id_mensajes = id_mensajes;
    }

    public String getFk_maestros() {
        return fk_maestros;
    }

    public void setFk_maestros(String fk_maestros) {
        this.fk_maestros = fk_maestros;
    }

    public int getFk_grupos() {
        return fk_grupos;
    }

    public void setFk_grupos(int fk_grupos) {
        this.fk_grupos = fk_grupos;
    }

    public String getTexto() {
        return texto;
    }

    public void setTexto(String texto) {
        this.texto = texto;
    }

    public String getImagen_url() {
        return imagen_url;
    }

    public void setImagen_url(String imagen_url) {
        this.imagen_url = imagen_url;
    }

    public Timestamp getFecha_envio() {
        return fecha_envio;
    }

    public void setFecha_envio(Timestamp fecha_envio) {
        this.fecha_envio = fecha_envio;
    }
}
```

DAO (Data Access Object)

Es un patrón de diseño estructural que se utiliza para separar la lógica de acceso a datos de la lógica de negocio. Los métodos DAO son responsables de interactuar con la base de datos (o cualquier otra fuente de datos) para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre las entidades del sistema. Este patrón es muy útil porque permite un mayor modularidad, reutilización del código y facilita el mantenimiento de la aplicación.

Estructura básica de un DAO

Normalmente, un DAO tiene los siguientes componentes:

Entidad (Java bean): Representa el objeto que se almacena en la base de datos.

Clase DAO: Define las operaciones básicas (CRUD) que se pueden realizar sobre la entidad.

Implementación: En este caso la implementación es en las interfaces del servidor RMI que se encarga de su utilización junto con la conexión de los Servlets.

En las imágenes de la siguiente página se veran los Métodos existentes en este proyecto:

Comenzando por uno de los mas importantes que sería el método de conexión que haciendo uso del controlador de MySQL puede hacer posible que se conecte Java a la BD

```
● ● ●

package Datos;

import java.sql.*;

public class Conexion {

    private static final String BD = " proyecto_redes";
    private static final String URL = "jdbc:mysql://localhost:3306/" + BD;
    private static final String USER = "root";
    private static final String PASSWORD = "admin";


    public static Connection getConexion() {
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Conexion exitosa");
        } catch (SQLException e) {
            System.out.println("Error al intentar conectarse a la base de datos: " + e);
        }
        return connection;
    }

    public static void cerrarConexion(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Conexion cerrada correctamente.");
            } catch (SQLException e) {
                System.err.println("Error al cerrar la conexion: " + e.getMessage());
            }
        }
    }

    public static void cerrarStatement(Statement statement) {
        if (statement != null) {
            try {
                statement.close();
                System.out.println("Statement cerrado correctamente.");
            } catch (SQLException e) {
                System.err.println("Error al cerrar el Statement: " + e.getMessage());
            }
        }
    }

    public static void cerrarResultSet(ResultSet resultSet) {
        if (resultSet != null) {
            try {
                resultSet.close();
                System.out.println("ResultSet cerrado correctamente.");
            } catch (SQLException e) {
                System.err.println("Error al cerrar el ResultSet: " + e.getMessage());
            }
        }
    }

}
```

En la clase Main se inicia el servidor RMI para hacer uso de los métodos del de las clases DAO

```
● ● ●

package Datos;

import Servicios.RegistroService;
import Servicios.impl.LoginServiceImpl;
import Servicios.impl.RegistroServiceImpl;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class Main {

    public static void main(String[] args) {
        try {
            // Establece el puerto y el hostname
            System.setProperty("java.rmi.server.hostname", "localhost");
            LocateRegistry.createRegistry(1099);

            RegistroService registroService = new RegistroServiceImpl();
            Naming.rebind("rmi://localhost:1099/RegistroService", registroService);

            LoginServiceImpl loginService = new LoginServiceImpl();
            Naming.rebind("rmi://localhost:1099/ServicioLogin", loginService);

            System.out.println("Servidor RMI iniciado correctamente");

        } catch (Exception e) {
            System.err.println("Error al iniciar el servidor RMI: " + e.getMessage());
        }
    }
}
```

AlumnosDAO se encarga de mantener métodos que ocupamos en el servidor RMI para poder cargar los datos de la pestaña de perfil.

```
● ● ●

package Datos;

import Modelo.Alumnos;
import java.sql.*;

public class AlumnosDAO {

    private static final String INSERT_ALUMNO = "INSERT INTO alumnos (matricula, nombre, segundo_nombre, apellido_paterno, apellido_materno, correo, contrasena) VALUES (?, ?, ?, ?, ?, ?, ?)";
    private static final String SELECT_ALUMNO_BY_MATRICULA = "SELECT * FROM alumnos WHERE matricula = ?";

    public Alumnos obtenerAlumnoPorMatricula(String matricula) {
        Connection connection = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Alumnos alumno = null;
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(SELECT_ALUMNO_BY_MATRICULA);
            ps.setString(1, matricula);
            rs = ps.executeQuery();
            if (rs.next()) {
                alumno = new Alumnos(
                    rs.getString("matricula"),
                    rs.getString("nombre"),
                    rs.getString("segundo_nombre"),
                    rs.getString("apellido_paterno"),
                    rs.getString("apellido_materno"),
                    rs.getString("correo"),
                    rs.getString("contrasena")
                );
            }
        } catch (SQLException e) {
            System.out.println("Error al obtener alumno: " + e.getMessage());
        } finally {
            Conexion.cerrarResultSet(rs);
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
        return alumno;
    }

    public int insertarAlumno(Alumnos alumno) {
        Connection connection = null;
        PreparedStatement ps = null;
        int resultado = 0;
        try {
            if (obtenerAlumnoPorMatricula(alumno.getMatricula()) == null) {
                connection = Conexion.getConexion();
                ps = connection.prepareStatement(INSERT_ALUMNO);
                ps.setString(1, alumno.getMatricula());
                ps.setString(2, alumno.getNombre());
                ps.setString(3, alumno.getSegundo_nombre());
                ps.setString(4, alumno.getApellido_paterno());
                ps.setString(5, alumno.getApellido_materno());
                ps.setString(6, alumno.getCorreo());
                ps.setString(7, alumno.getContrasena());
                ps.executeUpdate();
                resultado = 1;
            }
        } catch (SQLException e) {
            System.out.println("Error al insertar alumno: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
        return resultado;
    }
}
```

Grupos_Alumnos contiene métodos necesarios para por ejemplo para verificar si un alumno se encuentra ya registrado en el grupo y evitar su nueva inserción

```
● ● ●

package Datos;

import Modelo.Alumnos;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class Grupos_AlumnosDAO {

    private static final String INSERT_GRUPO_ALUMNO = "INSERT INTO grupos_alumnos (fk_grupos, fk_alumnos, fk_maestros) VALUES (?, ?, ?)";
    private static final String SELECT_BY_ALUMNO = "SELECT * FROM grupos_alumnos WHERE fk_alumnos = ? and fk_grupos=?";
    private static final String SELECT_ALUMNOS POR GRUPO =
        "SELECT a.matricula, a.nombre, a.segundo_nombre, a.apellido_paterno, a.apellido_materno,
a.correo " +
        "FROM grupos_alumnos ga " +
        "JOIN alumnos a ON ga.fk_alumnos = a.matricula " +
        "WHERE ga.fk_grupos = ?";

    private static final String DELETE_ALUMNO_DEL_GRUPO =
        "DELETE FROM grupos_alumnos WHERE fk_alumnos = ? AND fk_grupos = ?";

    private static final String COUNT_ALUMNOS_EN_GRUPO =
        "SELECT COUNT(*) FROM grupos_alumnos WHERE fk_alumnos = ? AND fk_grupos = ?";

    private static final String COUNT_MAESTRO_EN_GRUPO =
        "SELECT COUNT(*) FROM grupos_alumnos WHERE fk_maestros = ? AND fk_grupos = ?";

    public int contarMaestrosEnGrupo(String matricula, int grupo) {
        Connection connection = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        int count = 0;

        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(COUNT_MAESTRO_EN_GRUPO);
            ps.setString(1, matricula);
            ps.setInt(2, grupo);
            rs = ps.executeQuery();

            if (rs.next()) {
                count = rs.getInt(1);
            }
        } catch (SQLException e) {
            System.out.println("Error al contar los Maestros en el grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarResultSet(rs);
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }

        return count;
    }

    public int contarAlumnosEnGrupo(String matricula, int grupo) {
        Connection connection = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        int count = 0;

        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(COUNT_ALUMNOS_EN_GRUPO);
            ps.setString(1, matricula);
            ps.setInt(2, grupo);
            rs = ps.executeQuery();

            if (rs.next()) {
                count = rs.getInt(1);
            }
        } catch (SQLException e) {
            System.out.println("Error al contar los alumnos en el grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarResultSet(rs);
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }

        return count;
    }
}
```

```

public boolean eliminarAlumnoDeGrupo(String matricula, int grupo) {
    Connection connection = null;
    PreparedStatement ps = null;
    boolean eliminado = false;

    try {
        connection = Conexion.getConexion();
        ps = connection.prepareStatement(DELETE_ALUMNO_DEL_GRUPO);
        ps.setString(1, matricula);
        ps.setInt(2, grupo);
        ps.executeUpdate();
        eliminado = true;
    } catch (SQLException e) {
        System.out.println("Error al eliminar el alumno del grupo: " + e.getMessage());
    } finally {
        Conexion.cerrarStatement(ps);
        Conexion.cerrarConexion(connection);
    }

    return eliminado;
}

public List<Alumnos> obtenerTodosLosAlumnosPorGrupo(int grupo) {
    Connection connection = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    List<Alumnos> listaAlumnos = new ArrayList<>();
    try {
        connection = Conexion.getConexion();
        ps = connection.prepareStatement(SELECT_ALUMNOS_POR_GRUPO);
        ps.setInt(1, grupo);
        rs = ps.executeQuery();

        while (rs.next()) {
            Alumnos alumno = new Alumnos(
                rs.getString("matricula"),
                rs.getString("nombre"),
                rs.getString("segundo_nombre"),
                rs.getString("apellido_paterno"),
                rs.getString("apellido_materno"),
                rs.getString("correo"),
                ""
            );
            listaAlumnos.add(alumno);
        }
    } catch (SQLException e) {
        System.out.println("Error al obtener todos los alumnos: " + e.getMessage());
    } finally {
        Conexion.cerrarResultSet(rs);
        Conexion.cerrarStatement(ps);
        Conexion.cerrarConexion(connection);
    }
    return listaAlumnos;
}

public int obtenerGruposPorAlumno(String matricula, int idGrupo) {
    Connection connection = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    int cont = 0;

    try {
        connection = Conexion.getConexion();
        ps = connection.prepareStatement(SELECT_BY_ALUMNO);
        ps.setString(1, matricula);
        ps.setInt(2, idGrupo);
        rs = ps.executeQuery();

        while (rs.next()) {
            cont++;
        }
    } catch (SQLException e) {
        System.out.println("Error al obtener grupos por alumno: " + e.getMessage());
    } finally {
        Conexion.cerrarResultSet(rs);
        Conexion.cerrarStatement(ps);
        Conexion.cerrarConexion(connection);
    }
    return cont;
}

```

```
● ● ●

public void insertarGrupoAlumno(int idgrupo, String matricula, String ncontrol) {
    Connection connection = null;
    PreparedStatement ps = null;
    try {
        connection = Conexion.getConexion();
        ps = connection.prepareStatement(INSERT_GRUPO_ALUMNO);
        ps.setInt(1, idgrupo);
        ps.setString(2, matricula);
        ps.setString(3, ncontrol);
        ps.executeUpdate();
    } catch (SQLException e) {
        System.out.println("Error al insertar grupo-alumno: " + e.getMessage());
    } finally {
        Conexion.cerrarStatement(ps);
        Conexion.cerrarConexion(connection);
    }
}
```

Métodos:

1. insertarGrupoAlumno: se encarga de agregar un alumno a un grupo por parte del maestro.
2. obtenerGruposPorAlumno: se encarga de verificar si es que el alumno está inscrito en algún grupo, importante para la verificación si agregar alumno o no.
3. obtenerTodosLosAlumnosPorGrupo: se encarga de listar los alumnos que están en un determinado grupo, con ello hace posible el apartado de ver alumno, dentro del apartado del chat grupal.
4. eliminarAlumnoDeGrupo: se encarga de hacer posible sacar un alumno de dicho grupo, ya sea por error del profesor o el caso que sea.
5. contarAlumnosEnGrupo: Maneja la lógica del buscar alumnos para agregar a un grupo, siendo el caso que si ya esta no lo encuentra y no cambia la tabla de búsqueda.

GruposDAO gestiona la utilidad de los grupos, su creación, modificación y “eliminación” o cambio de estado.

```
● ● ●

package Datos;

import java.sql.*;

public class GruposDAO {

    private static final String INSERT_GRUPO = "INSERT INTO grupos (nombre, fk_maestros) VALUES (?,?)";
    private static final String UPDATE_GRUPO = "UPDATE grupos SET nombre = ? WHERE id_grupos = ?";
    private static final String DESACTIVAR_GRUPO = "UPDATE grupos SET estado = false WHERE id_grupos = ?";

    public void insertarGrupo(String nombre, String fkMaestro) {
        Connection connection = null;
        PreparedStatement ps = null;
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(INSERT_GRUPO);
            ps.setString(1, nombre);
            ps.setString(2, fkMaestro);
            ps.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Error al insertar grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
    }

    public void actualizarNombre(String nombre, int idgrupo) {
        Connection connection = null;
        PreparedStatement ps = null;
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(UPDATE_GRUPO);
            ps.setString(1, nombre);
            ps.setInt(2, idgrupo);
            ps.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Error al actualizar grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
    }

    public void desactivarGrupo(int id_grupos) {
        Connection connection = null;
        PreparedStatement ps = null;
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(DESACTIVAR_GRUPO);
            ps.setInt(1, id_grupos);
            ps.executeUpdate();
        } catch (SQLException e) {
            System.out.println("Error al desactivar grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
    }
}
```

MaestrosDAO se encarga de los métodos de registro en caso de ser un maestro distinguiéndose por ocupar un ID que no comienza en ZS como los alumnos y el método obtenerMaestroPorNControl se ocupa en algunas vistas para obtener su nombre y mostrarlo en pantalla

```
● ● ●

package Datos;

import Modelo.Maestros;

import java.sql.*;

public class MaestrosDAO {

    private static final String INSERT_MAESTRO = "INSERT INTO maestros (n_control, nombre, segundo_nombre, apellido_paterno, apellido_materno, correo, contrasena) VALUES (?, ?, ?, ?, ?, ?, ?)";
    private static final String SELECT_MAESTRO_BY_NCONTROL = "SELECT * FROM maestros WHERE n_control = ?";

    public Maestros obtenerMaestroPorNControl(String nControl) {
        Connection connection = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Maestros maestro = null;
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(SELECT_MAESTRO_BY_NCONTROL);
            ps.setString(1, nControl);
            rs = ps.executeQuery();
            if (rs.next()) {
                maestro = new Maestros(
                    rs.getString("n_control"),
                    rs.getString("nombre"),
                    rs.getString("segundo_nombre"),
                    rs.getString("apellido_paterno"),
                    rs.getString("apellido_materno"),
                    rs.getString("correo"),
                    rs.getString("contrasena")
                );
            }
        } catch (SQLException e) {
            System.out.println("Error al obtener maestro: " + e.getMessage());
        } finally {
            Conexion.cerrarResultSet(rs);
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
        return maestro;
    }

    public int insertarMaestro(Maestros maestro) {
        Connection connection = null;
        PreparedStatement ps = null;
        int veri = 0;
        try {
            if (obtenerMaestroPorNControl(maestro.getN_control()) == null) {
                connection = Conexion.getConexion();
                ps = connection.prepareStatement(INSERT_MAESTRO);
                ps.setString(1, maestro.getN_control());
                ps.setString(2, maestro.getNombre());
                ps.setString(3, maestro.getSegundo_nombre());
                ps.setString(4, maestro.getApellido_paterno());
                ps.setString(5, maestro.getApellido_materno());
                ps.setString(6, maestro.getCorreo());
                ps.setString(7, maestro.getContrasena());
                ps.executeUpdate();
                veri = 1;
            }
        } catch (SQLException e) {
            System.out.println("Error al insertar maestro: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
        return veri;
    }
}
```

Mensajes DAO es de las mas importantes ya que maneja el control de ellos, haciendo que puedan enviarse imágenes, imágenes y texto o texto, guardando la URL de dicha imagen que se guarda en el servidor (una carpeta dentro del proyecto)

```
● ● ●
package Datos;

import Modelo.Mensajes;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class MensajesDAO {

    private static final String INSERT_MENSAJE = "INSERT INTO mensajes (fk_maestros, fk_grupos, texto, imagen_url, fecha_envio) VALUES (?, ?, ?, ?, ?);";
    private static final String SELECT_MENSAJES POR_GRUPO = "SELECT * FROM mensajes WHERE fk_grupos = ? ORDER BY fecha_envio DESC";

    public List<Mensajes> obtenerMensajesPorGrupo(int idGrupo) {
        Connection connection = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        List<Mensajes> listaMensajes = new ArrayList<>();
        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(SELECT_MENSAJES POR_GRUPO);
            ps.setInt(1, idGrupo);
            rs = ps.executeQuery();

            while (rs.next()) {
                Mensajes mensaje = new Mensajes(
                    rs.getInt("id_mensajes"),
                    rs.getString("fk_maestros"),
                    rs.getInt("fk_grupos"),
                    rs.getString("texto"),
                    rs.getString("imagen_url"),
                    rs.getTimestamp("fecha_envio")
                );
                listaMensajes.add(mensaje);
            }
        } catch (SQLException e) {
            System.out.println("Error al obtener mensajes por grupo: " + e.getMessage());
        } finally {
            Conexion.cerrarResultSet(rs);
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
        return listaMensajes;
    }

    public void insertarMensaje(Mensajes mensaje) {
        Connection connection = null;
        PreparedStatement ps = null;
        boolean verificacion = false;

        try {
            connection = Conexion.getConexion();
            ps = connection.prepareStatement(INSERT_MENSAJE);
            ps.setInt(1, mensaje.getFk_maestros());
            ps.setInt(2, mensaje.getFk_grupos());

            // Si el texto es nulo se inserta como null
            if (mensaje.getTexto() != null && !mensaje.getTexto().isEmpty()) {
                ps.setString(3, mensaje.getTexto());
                verificacion = true;
            } else {
                ps.setNull(3, Types.VARCHAR);
            }

            // Si la imagen es nula se inserta como null
            if (mensaje.getImagen_url() != null && !mensaje.getImagen_url().isEmpty()) {
                ps.setString(4, mensaje.getImagen_url());
                verificacion = true;
            } else {
                ps.setNull(4, Types.VARCHAR);
            }

            ps.setTimestamp(5, mensaje.getFecha_envio());

            if (verificacion) {
                ps.executeUpdate();
            } else {
                System.out.println("\nNo se puede mandar mensaje sin texto o imagen");
            }
        } catch (SQLException e) {
            System.out.println("Error al insertar mensaje: " + e.getMessage());
        } finally {
            Conexion.cerrarStatement(ps);
            Conexion.cerrarConexion(connection);
        }
    }
}
```

TCPsockets al momento de implementarla para las pruebas y seguir las indicaciones de uso de TCP y UDP nos percatamos que al hacerlo en un navegador web nuestra aplicación había un problema, los sockets TCP y UDP de java nativos no son compatibles con los navegadores, aun así, dejamos el código.

```
● ● ●

package Datos;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class TCPsockets {
    /*private static List<GestorCliente> clientesConectados = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(9999);
        System.out.println("Servidor de chat listo en el puerto 9999...");
        while (true) {
            Socket socket = serverSocket.accept();
            GestorCliente ch = new GestorCliente(socket);
            synchronized (clientesConectados) {
                clientesConectados.add(ch);
            }
            ch.start();
        }
    }

    public static void broadcastMessage(String mensaje, int idGrupo) {
        synchronized (clientesConectados) {
            for (ClientHandler ch : clientesConectados) {
                if (ch.getIdGrupo() == idGrupo) {
                    ch.enviar(mensaje);
                }
            }
        }
    }*/
}

class GestorCliente extends Thread {
    private Socket socket;
    private PrintWriter out;
    private BufferedReader in;
    private int idGrupo;

    public GestorCliente(Socket socket) throws IOException {
        this.socket = socket;
        this.out = new PrintWriter(socket.getOutputStream(), true);
        this.in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    }

    public int getIdGrupo() {
        return idGrupo;
    }

    public void enviar(String msg) {
        out.println(msg);
    }

    @Override
    public void run() {
        try {
            String line = in.readLine();
            if (line != null) {
                this.idGrupo = Integer.parseInt(line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

WebSocketsCHAT fue la solución que encontramos, era la forma de implementar el chat en tiempo real de manera adecuada en un navegador, siendo compatibles nativamente, esta clase se encarga de que al momento donde un maestro envía un mensaje, aparte de insertarlo en la BD el alumno en dado caso de estar conectado a un websocket recibe las actualizaciones con el mensaje escrito. Logrando la comunicación de varios usuarios.

```

● ● ●

package Datos;

import javax.websocket.*;
import javax.websocket.server.ServerEndpoint;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

@ServerEndpoint("/wsChat")
public class webSocketsCHAT {
    private static Map<Integer, List<Session>> grupoSesiones = new ConcurrentHashMap<>();

    @OnOpen
    public void onOpen(Session session) {
        System.out.println("WebSocket abierto: " + session.getId());

        String query = session.getQueryString();
        int idGrupo = parsearIdGrupo(query);

        if (idGrupo != -1) {
            grupoSesiones.computeIfAbsent(idGrupo, k -> new CopyOnWriteArrayList<>()).add(session);
        } else {
            System.out.println("idGrupo invalido: " + query);
        }
    }

    @OnMessage
    public void onMessage(String message, Session session) {
    }

    @OnClose
    public void onClose(Session session) {
        String query = session.getQueryString();
        int idGrupo = parsearIdGrupo(query);
        if (idGrupo != -1) {
            List<Session> sesiones = grupoSesiones.get(idGrupo);
            if (sesiones != null) {
                sesiones.remove(session);
                if (sesiones.isEmpty()) {
                    grupoSesiones.remove(idGrupo);
                }
            }
        }
        System.out.println("WebSocket cerrado: " + session.getId());
    }

    @OnError
    public void onError(Session session, Throwable throwable) {
        System.out.println("Error en WebSocket: " + session.getId() + " - " + throwable.getMessage());
    }

    public static void enviarMensajeAGrupo(int idGrupo, String mensaje) {
        List<Session> sesiones = grupoSesiones.get(idGrupo);
        if (sesiones != null) {
            for (Session s : sesiones) {
                if (s.isOpen()) {
                    s.getAsyncRemote().sendText(mensaje);
                }
            }
        }
    }

    private int parsearIdGrupo(String query) {
        if (query != null && query.startsWith("idGrupo=")) {
            try {
                return Integer.parseInt(query.substring("idGrupo=".length()));
            } catch (NumberFormatException e) {
                System.out.println("numero de grupo invalido en query " + query);
            }
        }
        return -1;
    }
}

```

RMI

Al empezar el proyecto con la investigación que había que hacer, no se tuvo en cuenta que el servidor RMI podía soportar toda la variedad de métodos en el DAO, por lo que se optó por primero, crear uno para lo del login y otro para registro, después con mayor conocimiento de ellos, se ocupó como principal la interfaz de LoginService:

```
● ● ●

package Servicios;

import Modelo.*;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface LoginService extends Remote {
    boolean loginAlumno(String matricula, String contrasena) throws RemoteException;

    boolean loginMaestro(String numeroControl, String contrasena) throws RemoteException;

    String obtenerNombreAlumno(String matricula) throws RemoteException;

    String obtenerNombreMaestro(String numeroControl) throws RemoteException;

    List<Grupos_Alumnos> obtenerGruposPorAlumno(String fk_alumnos) throws RemoteException;

    List<Grupos> obtenerGruposPorMaestro(String fk_maestros) throws RemoteException;

    Alumnos obtenerAlumnoPorMatricula(String matricula) throws RemoteException;

    Maestros obtenerMaestroPorNControl(String nControl) throws RemoteException;

    void insertarGrupoAlumno(int idgrupo, String matricula, String ncontrol) throws RemoteException;

    int obtenerGruposPorAlumno(String matricula, int idGrupo) throws RemoteException;

    void insertarGrupo(String nombre, String fkMaestro) throws RemoteException;

    void desactivarGrupo(int id_grupos) throws RemoteException;

    void actualizarNombre(String nombre, int idgrupo) throws RemoteException;

    List<Alumnos> obtenerTodosLosAlumnosPorGrupo(int grupo) throws RemoteException;

    boolean eliminarAlumnoDeGrupo(String matricula, int grupo) throws RemoteException;

    void enviarMensaje(Mensajes mensaje) throws RemoteException;

    List<Mensajes> obtenerMensajesPorGrupo(int idGrupo) throws RemoteException;

    int contarAlumnosEnGrupo(String matricula, int grupo) throws RemoteException;

    int contarMaestrosEnGrupo(String matricula, int grupo) throws RemoteException;
}
```

La implementación de los métodos RMI se haría en la clase que implementa la interfaz de LoginService siendo la clase “LoginServiceImpl”, aquí es donde se implementan los métodos de las clases DAO de manera remota de ser el caso y usar un servidor diferente a localhost.

```
● ● ●

package Servicios.impl;

import Datos.*;
import Modelo.*;
import Servicios.LoginService;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class LoginServiceImpl extends UnicastRemoteObject implements LoginService {

    public LoginServiceImpl() throws Exception {
        super();
    }

    @Override
    public Alumnos obtenerAlumnoPorMatricula(String matricula) {
        AlumnosDAO alumnosDAO = new AlumnosDAO();
        return alumnosDAO.obtenerAlumnoPorMatricula(matricula);
    }

    @Override
    public Maestros obtenerMaestroPorNControl(String nControl) {
        MaestrosDAO maestrosDAO = new MaestrosDAO();
        return maestrosDAO.obtenerMaestroPorNControl(nControl);
    }

    @Override
    public void insertarGrupoAlumno(int idgrupo, String matricula, String ncontrol) {
        Grupos_AlumnosDAO gruposDAO = new Grupos_AlumnosDAO();
        gruposDAO.insertarGrupoAlumno(idgrupo, matricula, ncontrol);
    }

    @Override
    public int obtenerGruposPorAlumno(String matricula, int idGrupo) {
        Grupos_AlumnosDAO gruposDAO = new Grupos_AlumnosDAO();
        return gruposDAO.obtenerGruposPorAlumno(matricula, idGrupo);
    }

    @Override
    public void insertarGrupo(String nombre, String fkMaestro) {
        GruposDAO gruposDAO = new GruposDAO();
        gruposDAO.insertarGrupo(nombre, fkMaestro);
    }

    @Override
    public void desactivarGrupo(int id_grupos) {
        GruposDAO gruposDAO = new GruposDAO();
        gruposDAO.desactivarGrupo(id_grupos);
    }

    @Override
    public void actualizarNombre(String nombre, int idgrupo) {
        GruposDAO gruposDAO = new GruposDAO();
        gruposDAO.actualizarNombre(nombre, idgrupo);
    }

    @Override
    public List<Alumnos> obtenerTodosLosAlumnosPorGrupo(int grupo) {
        Grupos_AlumnosDAO gruposDAO = new Grupos_AlumnosDAO();
        return gruposDAO.obtenerTodosLosAlumnosPorGrupo(grupo);
    }

    @Override
    public boolean eliminarAlumnoDeGrupo(String matricula, int grupo) {
        Grupos_AlumnosDAO gruposDAO = new Grupos_AlumnosDAO();
        return gruposDAO.eliminarAlumnoDeGrupo(matricula, grupo);
    }

    @Override
    public void enviarMensaje(Mensajes mensaje) {
        MensajesDAO mensajesDAO = new MensajesDAO();
        mensajesDAO.insertarMensaje(mensaje);
    }
}
```

Trabajando como se comentó por desconocimiento con 2 servidores RMI, siendo el otro RegistroService y su clase que implemente RegistroServiceImpl

```
● ● ●

package Servicios;

import Modelo.Alumnos;
import Modelo.Maestros;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface RegistroService extends Remote {
    boolean registrarAlumno(Alumnos alumno) throws RemoteException;
    boolean registrarMaestro(Maestros maestro) throws RemoteException;
}
```

```
● ● ●

package Servicios.impl;

import Datos.AlumnosDAO;
import Datos.MaestrosDAO;
import Modelo.Alumnos;
import Modelo.Maestros;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class RegistroServiceImpl extends UnicastRemoteObject implements Servicios.RegistroService {

    public RegistroServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public boolean registrarAlumno(Alumnos alumno) throws RemoteException {
        AlumnosDAO dao = new AlumnosDAO();
        if (dao.insertarAlumno(alumno) == 1) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public boolean registrarMaestro(Maestros maestro) throws RemoteException {
        MaestrosDAO dao = new MaestrosDAO();
        if (dao.insertarMaestro(maestro) == 1) {
            return true;
        } else {
            return false;
        }
    }
}
```

Servlets

Los servlets son una de las tecnologías fundamentales en el desarrollo de aplicaciones web en Java, ya que permiten la comunicación entre las vistas (como JSP o tecnologías front-end como JavaScript) y la lógica de negocio o la capa de persistencia (bases de datos, APIs, etc.). Actúan como intermediarios entre las solicitudes de los usuarios y la lógica que maneja los datos, permitiendo crear aplicaciones web dinámicas.

Función de los Servlets

El servlet se encarga de manejar las solicitudes HTTP recibidas del cliente (navegador), procesarlas y devolver una respuesta adecuada. Esto incluye:

1. Interactuar con las bases de datos o servicios backend.
2. Realizar cálculos o procesamiento de datos.
3. Generar contenido dinámico (HTML, JSON, XML, etc.).
4. Manejar la navegación de la aplicación.

Servlets ocupados en el proyecto

Mostraremos un listado de los que se hicieron para completar las funcionalidades del proyecto:

1. AgregarAlumnoGrupoServlet
2. BuscarAlumnoServlet
3. CerrarSesionServlet
4. CrearGrupoServlet
5. EliminarAlumnoGrupoServlet
6. EnviarMensajeServlet
7. LoginServlet
8. ModificarEliminarGrupoServlet
9. registerServlet

AgregarAlumnoServlet se utiliza en AgregarAlumnos.jsp donde cumple la labor de insertar el alumno siempre y cuando el id del grupo que se pase sea valido

```
● ● ●

package Controlador;

import Modelo.Alumnos;
import Servicios.LoginService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.rmi.Naming;

@WebServlet("/AgregarAlumnoGrupoServlet")
public class AgregarAlumnoGrupoServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String matricula = request.getParameter("matricula");
        String grupoStr = request.getParameter("id_grupos");
        String ncontrol = request.getParameter("ncontrol");
        String materia = request.getParameter("materia");
        String nombre = request.getParameter("nombre");
        System.out.println(nombre);

        if (grupoStr == null || grupoStr.isEmpty()) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST, "id de grupo obligatorio");
            return;
        }

        try {
            int grupo = Integer.parseInt(grupoStr);

            LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
            if (loginService.obtenerGruposPorAlumno(matricula, grupo) == 0) {
                loginService.insertarGrupoAlumno(grupo, matricula, ncontrol);
            } else {
                System.out.println("ya esta el alumno");
            }

            response.sendRedirect("ChatMaestro.jsp?materia=" + materia + "&id_grupos=" + grupo +
"&nombre=" + nombre);
        } catch (NumberFormatException e) {
            e.printStackTrace();
            response.sendError(HttpServletResponse.SC_BAD_REQUEST, "id de grupo invalido");
        } catch (Exception e) {
            e.printStackTrace();
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "Error al agregar el alumno:
" + e.getMessage());
        }
    }
}
```

BuscarAlumnoServlet es el servlet que se encarga de buscar los alumnos siempre y no estén en el grupo, devolviendo sus datos en base a su matrícula para así hacer posible agregarlos a un grupo:

```
● ● ●

package Controlador;

import Modelo.Alumnos;
import Servicios.LoginService;
import Servicios.RegistroService;
import Modelo.Grupos_Alumnos;

import java.io.IOException;
import java.rmi.Naming;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/BuscarAlumnoServlet")
public class BuscarAlumnoServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String matricula = request.getParameter("busqueda");
        String materia = request.getParameter("materia");
        String id_grupos = request.getParameter("id_grupos");
        String nombre = request.getParameter("nombre");
        String mensaje = "";
        Alumnos alumno = null;

        try {
            LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
            if (loginService.contarAlumnosEnGrupo(matricula, Integer.parseInt(id_grupos)) == 0) {
                alumno = loginService.obtenerAlumnoPorMatricula(matricula);
            }else{
                mensaje = "El alumno ya esta en el grupo";
            }
        } catch (Exception e) {
            e.printStackTrace();
            request.setAttribute("mensaje", "Error al buscar el alumno: " + e.getMessage());
        }

        request.setAttribute("resultadosBusqueda", alumno);
        request.setAttribute("materia", materia);
        request.setAttribute("id_grupos", id_grupos);
        request.setAttribute("nombre", nombre);
        request.setAttribute("mensaje", mensaje);

        request.getRequestDispatcher("AgregarAlumnos.jsp").forward(request, response);
    }
}
```

CerrarSesionServlet se encarga en todas las vistas de hacer posible el cierre de sesión anulando la existente, todas las vistas (JSP) están validadas para no hacer posible el acceso si no se cuenta con sesión previamente valida

```
● ● ●

package Controlador;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/CerrarSesionServlet")
public class CerrarSesionServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        if (request.getSession(false) != null) {
            request.getSession().invalidate();
        }
        response.sendRedirect("index.jsp");
    }
}
```

CrearGrupoServlet se encarga de la creación de grupos en la vista CrearChats.jsp

```
● ● ●

package Controlador;

import Servicios.LoginService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.rmi.Naming;
import java.rmi.NotBoundException;

@WebServlet("/CrearGrupoServlet")
public class CrearGrupoServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String nombre = request.getParameter("nombreGrupo");
        String ncontrol = request.getParameter("ncontrol");

        try {
            LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
            loginService.insertarGrupo(nombre, ncontrol);
            response.sendRedirect("MenuMaestro.jsp");
        } catch (NumberFormatException e) {
            e.printStackTrace();
        } catch (NotBoundException e) {
            throw new RuntimeException(e);
        }
    }
}
```

EliminarAlumnoGrupoServlet se encarga en la vista VerMiembros.jsp de eliminar los alumnos de un determinado grupo

```
● ● ●

package Controlador;

import Servicios.LoginService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.rmi.Naming;

@WebServlet("/EliminarAlumnoGrupoServlet")
public class EliminarAlumnoGrupoServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String matricula = request.getParameter("matricula");
        String grupoStr = request.getParameter("id_grupos");
        String materia = request.getParameter("materia");
        String nombre = request.getParameter("nombre");

        int grupoint = 0;
        if (grupoStr != null && !grupoStr.equals("")) {
            grupoint = Integer.parseInt(grupoStr);
        }

        try {
            LoginService loginService = (LoginService)
            Naming.lookup("rmi://localhost:1099/ServicioLogin");
            loginService.eliminarAlumnoDeGrupo(matricula, grupoint);
            response.sendRedirect("ChatMaestro.jsp?materia=" + materia + "&id_grupos=" + grupoint+
            "&nombre=" + nombre);
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

ModificarEliminarGrupoServlet se encarga en la vista de ModificarEliminarChats de la parte de guardar los cambios realizados al titulo del grupo o la eliminación del grupo

```
● ● ●

package Controlador;

import Servicios.LoginService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.rmi.Naming;
import java.rmi.NotBoundException;

@WebServlet("/ModificarEliminarGrupoServlet")

public class ModificarEliminarGrupoServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String nombreGrupo = request.getParameter("nombre_grupo");
        String idGrupo = request.getParameter("id_grupo");
        String accion = request.getParameter("accion");
        if (accion.equals("eliminar")) {
            try {
                LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
                loginService.desactivarGrupo(Integer.parseInt(idGrupo));
                response.sendRedirect("MenuMaestro.jsp");
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else if (accion.equals("guardar")) {
            try {
                LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
                loginService.actualizarNombre(nombreGrupo, Integer.parseInt(idGrupo));
                response.sendRedirect("MenuMaestro.jsp");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

LoginServlet se encarga del inicio de sesión en la vista de index.jsp validando de manera exitosa que las credenciales sean correctas

```
● ● ●

package Controlador;

import Servicios.LoginService;
import java.io.IOException;
import java.rmi.Naming;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String usuario = request.getParameter("usuario");
        String contrasena = request.getParameter("contrasena");

        try {
            LoginService loginService = (LoginService)
Naming.lookup("rmi://localhost:1099/ServicioLogin");
            String nombreUsuario = null;

            if (usuario.toLowerCase().startsWith("zs")) {
                boolean esValido = loginService.loginAlumno(usuario, contrasena);
                if (esValido) {
                    nombreUsuario = loginService.obtenerNombreAlumno(usuario);
                    request.getSession().setAttribute("usuario", usuario);
                    request.getSession().setAttribute("nombre", nombreUsuario);
                    response.sendRedirect("MenuAlumno.jsp");
                } else {
                    request.setAttribute("mensaje", "Credenciales incorrectas para alumno.");
                    request.getRequestDispatcher("index.jsp").forward(request, response);
                }
            } else {
                boolean esValido = loginService.loginMaestro(usuario, contrasena);
                if (esValido) {
                    nombreUsuario = loginService.obtenerNombreMaestro(usuario);
                    request.getSession().setAttribute("usuario", usuario);
                    request.getSession().setAttribute("nombre", nombreUsuario);
                    response.sendRedirect("MenuMaestro.jsp");
                } else {
                    request.setAttribute("mensaje", "Credenciales incorrectas para maestro.");
                    request.getRequestDispatcher("index.jsp").forward(request, response);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            request.setAttribute("mensaje", "Error al conectarse al servidor.");
            request.getRequestDispatcher("index.jsp").forward(request, response);
        }
    }
}
```

RegistroServlet se encarga de validar los campos ingresados en la vista de Registro.jsp para hacer uso de la plataforma

```
● ● ●

package Controlador;

import Modelo.Alumnos;
import Modelo.Maestros;
import Servicios.RegistroService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.rmi.Naming;

@WebServlet("/registerServlet")
public class RegistroServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String matricula = request.getParameter("matricula");
        String nombre = request.getParameter("nombre");
        String segundoNombre = request.getParameter("segundoNombre");
        String apellidoPaterno = request.getParameter("apellidoPaterno");
        String apellidoMaterno = request.getParameter("apellidoMaterno");
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String confirm_password = request.getParameter("confirm_password");
        String mensaje = "";

        RegistroService registroService;
        try {
            registroService = (RegistroService) Naming.lookup("rmi://localhost/RegistroService");
        } catch (Exception e) {
            throw new ServletException("No se pudo conectar al servicio RMI", e);
        }

        try {
            boolean registrado = false;
            if (confirm_password.equals(password)) {
                if (matricula.toLowerCase().startsWith("zs")) {
                    Alumnos alumno = new Alumnos(
                        matricula, nombre, segundoNombre, apellidoPaterno, apellidoMaterno, email,
password
                    );
                    registrado = registroService.registrarAlumno(alumno);
                } else {
                    Maestros maestro = new Maestros(
                        matricula, nombre, segundoNombre, apellidoPaterno, apellidoMaterno, email,
password
                    );
                    registrado = registroService.registrarMaestro(maestro);
                }
                mensaje = "Error: Matricula registrada";
            } else {
                mensaje = "Contraseñas no son iguales";
            }
            if (registrado) {
                response.sendRedirect("index.jsp?mensaje=Registro exitoso");
            } else {
                response.sendRedirect("Registro.jsp?error=" + mensaje);
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.sendRedirect("Registro.jsp?error=Error al conectar con el servidor");
        }
    }
}
```

EnviarMensajeServlet es el que hace lo posible en la vista de chatMaestro.jsp de hacer el envío de mensajes gracias a el método de enviar mensaje dentro de la clase de websocketsCHAT y en la base de datos gracias a la implementación RMI de enviarMensaje dentro de LoginService.

```
● ● ●
package Controlador;

import Datos.webSocketsCHAT;
import Modelo.Mensajes;
import Servicios.LoginService;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;
import java.rmi.Naming;
import java.sql.Timestamp;
import java.util.UUID;

@WebServlet("/EnviarMensajeServlet")
@MultipartConfig
public class EnviarMensajeServlet extends HttpServlet {

    private static final String IMAGENES_URL = "http://localhost:8080/ProyectoRedes/imagenes/";

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String fk_maestros = (String) request.getSession().getAttribute("usuario");
        String materia = request.getParameter("materia");
        String nombre = request.getParameter("nom");

        String texto = request.getParameter("mensaje");
        if (texto != null) {
            texto = texto.trim();
            if (texto.isEmpty()) {
                texto = null;
            }
        }

        String grupo = request.getParameter("grupo");
        if (grupo == null || grupo.trim().isEmpty()) {
            response.sendRedirect("ChatMaestro.jsp?id_grupos=" + grupo + "&materia=" + materia + "&nombre=" + nombre);
            return;
        }

        int fk_grupos = Integer.parseInt(grupo);

        String imagen_url = null;
        Part imagenPart = request.getPart("imagen");
        if (imagenPart != null && imagenPart.getSize() > 0) {
            String tipoArchivo = imagenPart.getContentType();
            if (tipoArchivo.startsWith("image/")) {
                String extension = getExtensionArchivo(getNombreArchivo(imagenPart));
                String nombreArchivo = UUID.randomUUID().toString() + extension;
                String direccionImagen = getServletContext().getRealPath("/imagenes/");

                File img = new File(direccionImagen);
                if (!img.exists()) {
                    img.mkdirs();
                }
                File imgExtension = new File(img, nombreArchivo);
                try (InputStream input = imagenPart.getInputStream()) {
                    Files.copy(input, imgExtension.toPath(), StandardCopyOption.REPLACE_EXISTING);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                imagen_url = IMAGENES_URL + nombreArchivo;
            } else {
                System.out.println("Tipo de archivo no permitido: " + tipoArchivo);
            }
        }

        if (texto == null && imagen_url == null) {
            response.sendRedirect("ChatMaestro.jsp?id_grupos=" + fk_grupos + "&materia=" + materia + "&nombre=" + nombre);
            return;
        }
    }
}
```

```
try {
    LoginService mensajeriaService = (LoginService) Naming.lookup("rmi://localhost:1099/ServicioLogin");
    Mensajes mensaje = new Mensajes();
    mensaje.setFk_maestros(fk_maestros);
    mensaje.setFk_grupos(fk_grupos);
    mensaje.setTexto(texto);
    mensaje.setImagen_url(imagen_url);
    mensaje.setFecha_envio(new Timestamp(System.currentTimeMillis()));

    mensajeriaService.enviarMensaje(mensaje);

    if (imagen_url != null) {
        String json = "[" +
            "\\"type\\":\"image\", " +
            "\\"text\\":\"" + formatoJSON(texto != null ? texto : "") + "\", " +
            "\\"imageData\\":\"" + imagen_url + "\" +
        "]";
        webSocketsCHAT.enviarMensajeAGrupo(fk_grupos, json);
    } else if (texto != null) {
        String json = "{" +
            "\\"type\\":\"text\", " +
            "\\"text\\":\"" + formatoJSON(texto) + "\" +
        "}";
        webSocketsCHAT.enviarMensajeAGrupo(fk_grupos, json);
    }
}

response.sendRedirect("ChatMaestro.jsp?id_grupos=" + fk_grupos + "&materia=" + materia + "&nombre=" + nombre);

} catch (Exception e) {
    e.printStackTrace();
}
}

private String getExtensionArchivo(String nombreArchivo) {
    if (nombreArchivo == null) return "";
    int index = nombreArchivo.lastIndexOf('.');
    return (index == -1) ? "" : nombreArchivo.substring(index);
}

private String getNombreArchivo(Part part) {
    String header = part.getHeader("content-disposition");
    if (header == null) return null;
    for (String contenido : header.split(";")) {
        if (contenido.trim().startsWith("filename")) {
            String nombreArchivo = contenido.substring(contenido.indexOf('=') + 1).trim().replace("\\", "/");
            return nombreArchivo;
        }
    }
    return null;
}

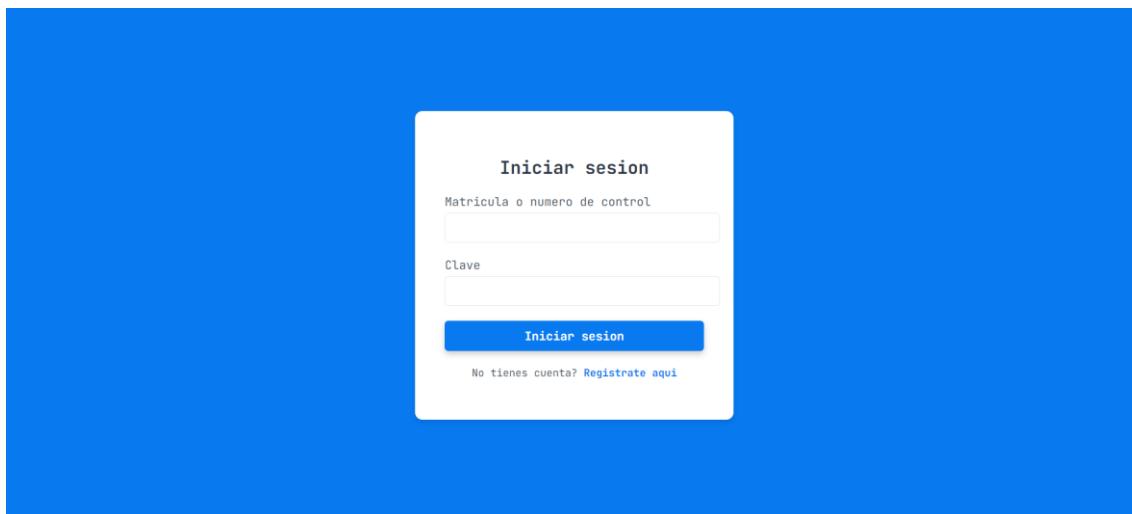
private String formatoJSON(String texto) {
    if (texto == null) return "";
    return texto.replace("\\", "\\\\")
        .replace("\n", "\\\\n")
        .replace("/", "\\\\/")
        .replace("\b", "\\\\b")
        .replace("\f", "\\\\f")
        .replace("\n", "\\\\n")
        .replace("\r", "\\\\r")
        .replace("\t", "\\\\t");
}
}
```

Gracias a este metodo podemos cargar con el metodo on.messages dentro de ChatAlumnos los datos que se acaban de enviar en tiempo real, haciendo posible que todos los alumnos reciban la informacion al momento.

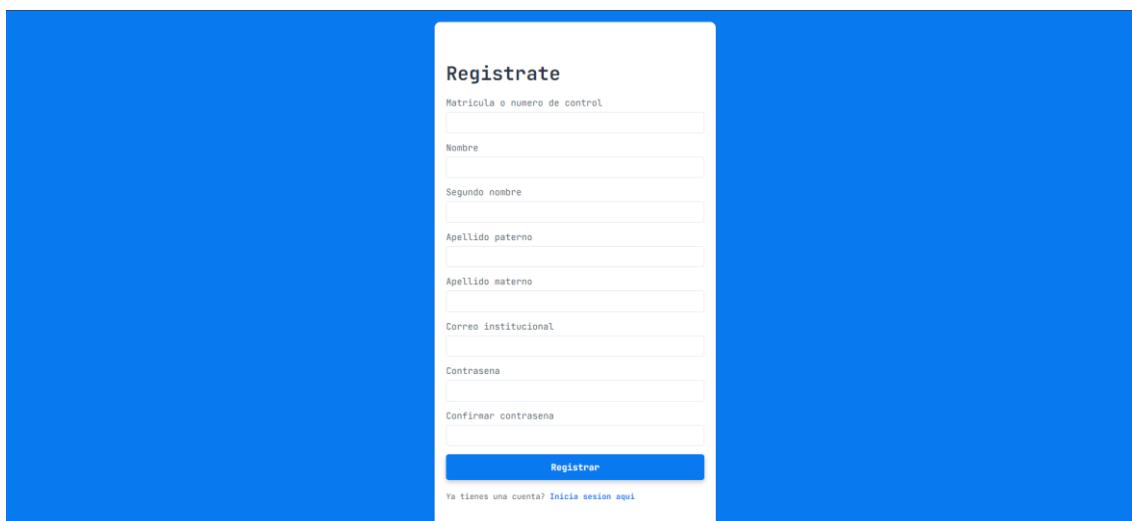
Resultados

Gracias a todo lo anterior pudimos concluir una aplicación de mensajería integrando los conceptos de los canales de WhatsApp, pero enfocado para el ámbito académico que tiene MS Teams, a continuación, se mostraran algunas imágenes del proyecto terminado:

Inicio de sesión



Registro



Menú para alumnos

The screenshot shows the student menu interface. On the left is a sidebar with a blue header "Teams UV" and a list of options: "Inicio", "Chats de trabajo", "Estructura", "Grupo maestro 2", "Perfil", and "Cerrar sesión". The "Chats de trabajo" section contains a green button labeled "-- CREAR CHAT --". The main content area has a white header "Bienvenido alumno: María". Below it is a box titled "Materias Inscritas" containing two items: "Estructura" and "Grupo maestro 2". Another box titled "Información" contains the text: "Con esta aplicación podrás estar al corriente de los avisos que los maestros realicen en cada una de sus materias".

Menú para maestros

The screenshot shows the teacher menu interface. On the left is a sidebar with a blue header "Teams UV" and a list of options: "Inicio", "Chats de trabajo", "Estructura", "MODIFICAR O ELIMINAR" (highlighted in green), "Perfil", and "Cerrar sesión". The "Chats de trabajo" section contains a green button labeled "-- CREAR CHAT --". The main content area has a white header "Bienvenido maestro: Carlos". Below it is a box titled "Gestiona tus grupos" containing the item "Estructura". Another box titled "Información" contains the text: "Este es tu espacio para mantenerte conectado con tus alumnos y gestionar materiales de estudio".

Perfil

Teams UV

- Inicio
- Perfil
- Cerrar sesión

Perfil de Carlos

Matrícula: 123
Nombre: Carlos Eduardo
Apellido Paterno: Gómez
Apellido Materno: López
Correo electrónico: carlos.gomez@example.com

Chat de alumnos

Teams UV

- Inicio
- Perfil
- Cerrar sesión

Chat de Estructura

Carlos: Imagen enviada



Carlos: aaa
Carlos: aaaa
Carlos: Imagen enviada



Solo lectura: No puedes escribir mensajes.

Chat de maestros

The screenshot shows a 'Chat de Estructura' window. On the left is a sidebar with 'Teams UV' and navigation links: 'Inicio', 'Perfil', and 'Cerrar sesión'. The main area displays a conversation with a user named 'Carlos'. The messages are:

- Carlos: Imagen enviada (with a small thumbnail image)
- Carlos: aaa
- Carlos: aaaa
- Carlos: Imagen enviada (with a large black redacted image)

At the bottom are 'Enviar' and 'Adjuntar' buttons, and a text input field 'Escribe un mensaje...'. Top right buttons are 'Agregar alumnos' and 'Ver alumnos'.

Agregar alumnos a un grupo (Maestro)

The screenshot shows a search interface for adding students to a group. The sidebar has the same 'Teams UV' and navigation links. The main area is titled 'Agregar alumnos en Estructura' with a 'Atras' button. It includes a search bar with 'Ingresa la matrícula...' and a 'Buscar' button. Below is a table titled 'Resultados de la búsqueda' with columns 'Matrícula', 'Nombre', and 'Acción'. One result is listed:

Matrícula	Nombre	Acción
zs2204	Martin Isabel Pérez Hernández	Agregar

Ver alumnos de un grupo (Maestro)

Teams UV

Inicio

Perfil

Cerrar sesión

Consulta los miembros de la clase

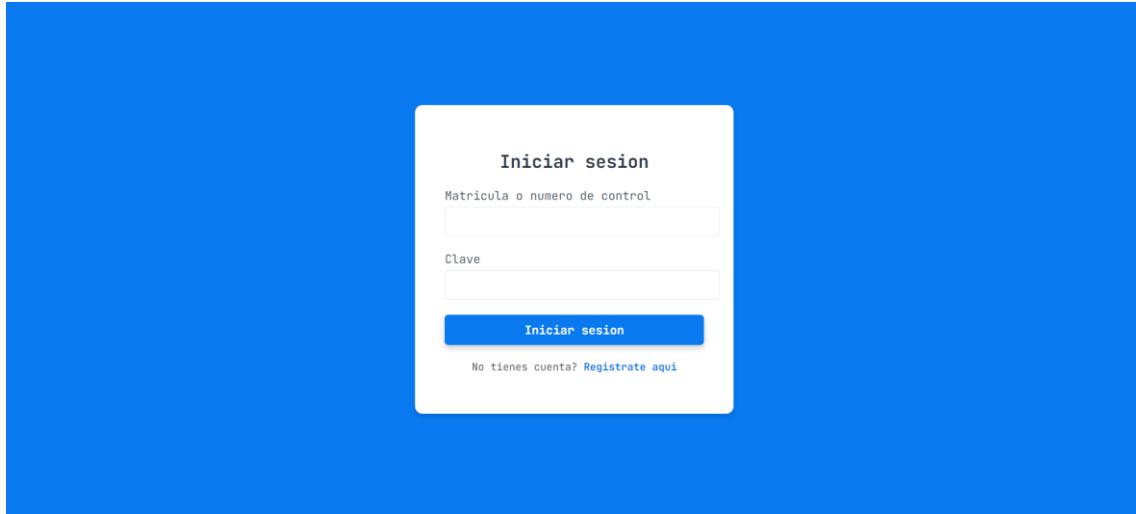
Atras

Alumnos del Grupo

Matrícula	Nombre	Correo	Eliminar del grupo
zs2282	Maria Isabel Pérez Hernández	maria.perez@example.com	Eliminar
zs2283	Gustavo Isabel Pérez Hernández	maria.perez@example.com	Eliminar

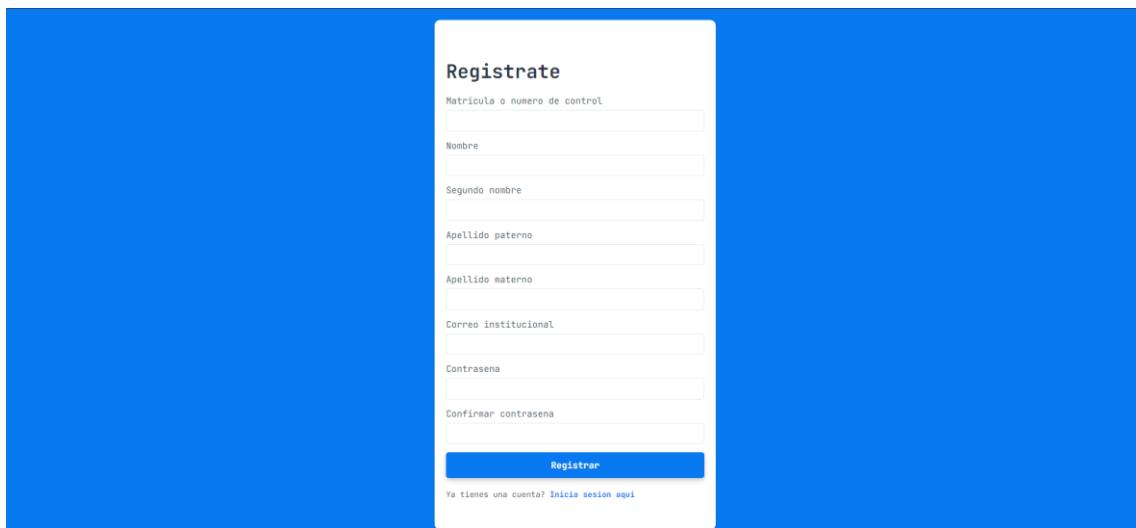
Instrucciones

Inicio de sesión



El usuario debe escribir su matrícula o número de control dependiendo si es maestro o alumno además de su contraseña y dar click en iniciar sesión, en caso de no estar registrado hacer click en “Registrate aquí” que redirigira a la vista de registro.

Registro



El usuario debe completar todos los campos dependiendo el caso de ser maestro o alumno y confirmar su contraseña, cuando tenga todos los datos escritos debe dar click en “Registrar”, si es que ya tiene una cuenta, puede regresar dando click en “Inicia sesión aquí” para ser redirigido al login.

Menú para alumnos

The screenshot shows the student menu interface. On the left, there is a vertical sidebar with a blue header labeled "Teams UV". Below the header, the sidebar contains the following menu items: "Inicio", "Chats de trabajo", "Estructura", "Grupo maestro 2", "Perfil", and "Cerrar sesión". The "Perfil" item is highlighted with a white background. To the right of the sidebar, the main content area has a light gray background. At the top, it says "Bienvenido alumno: María". Below this, there is a section titled "Materias Inscritas" which lists "Estructura" and "Grupo maestro 2". Another section titled "Información" contains the text: "Con esta aplicación podrás estar al corriente de los avisos que los maestros realicen en cada una de sus materias".

En caso de ser un alumno, tienes la posibilidad de ver tu perfil en la opción de la izquierda de “Perfil” o puedes ver las materias donde se está inscrito desplegando con un click el menú de “Chats de trabajo” y hacer click en el grupo que deseas checar para ser redirigido a su chat pertinente.

Menú para maestros

The screenshot shows the teacher menu interface. On the left, there is a vertical sidebar with a blue header labeled "Teams UV". Below the header, the sidebar contains the following menu items: "Inicio", "Chats de trabajo", "Estructura", and "MODIFICAR O ELIMINAR". The "MODIFICAR O ELIMINAR" item is highlighted with a green background. The other items ("Inicio", "Chats de trabajo", and "Estructura") have a light blue background. To the right of the sidebar, the main content area has a light gray background. At the top, it says "Bienvenido maestro: Carlos". Below this, there is a section titled "Gestiona tus grupos" which lists "Estructura". Another section titled "Información" contains the text: "Este es tu espacio para mantenerte conectado con tus alumnos y gestionar materiales de estudio."

Al ser maestro, tienes la posibilidad de ver tu perfil en la opción de la izquierda de “Perfil” o puedes ver las materias que has creado desplegando con un click el menú de “Chats de trabajo” y hacer click en el grupo que deseas checar para ser redirigido a su chat pertinente, también puedes ir a crear un chat dando click el apartado de “-- CREAR CHAT --” o puedes borrar el chat o modificar su nombre

dando click en el apartado de “MODIFICAR O ELIMINAR” estos últimos mostrados de un color verde.

Perfil

The screenshot shows the 'Perfil' (Profile) page. On the left, there is a sidebar with a blue header 'Teams UV' and three buttons: 'Inicio', 'Perfil', and 'Cerrar sesión'. The main content area has a title 'Perfil de Carlos' and a box containing personal information: Matrícula: 123, Nombre: Carlos Eduardo, Apellido Paterno: Gómez, Apellido Materno: López, and Correo electrónico: carlos.gomez@example.com.

En perfil el usuario puede ver sus datos personales además de poder volver al menú de inicio o cerrar sesión como únicas opciones

Chat de alumnos

The screenshot shows the 'Chat de Estructura' (Student Chat) page. On the left, there is a sidebar with a blue header 'Teams UV' and three buttons: 'Inicio', 'Perfil', and 'Cerrar sesión'. The main content area has a title 'Chat de Estructura' and a list of messages from 'Carlos': 'Imagen enviada' (with a small thumbnail image), 'aaa', 'aaaa', and another 'Imagen enviada' (with a larger thumbnail image). At the bottom, there is a message placeholder 'Solo lectura: No puedes escribir mensajes.'

En la vista de chat para alumnos el usuario puede ver los mensajes en manera descendente y dar click a las imágenes para ampliarlas, además de poder dar click en “Perfil” para ver el suyo o si lo requiere ir al inicio.

Chat de maestros

The screenshot shows a chat interface titled "Chat de Estructura". On the left is a sidebar with "Teams UV" and navigation links: "Inicio", "Perfil", and "Cerrar sesión". The main area has a header "Chat de Estructura" with "Agregar alumnos" and "Ver alumnos" buttons. The conversation log shows messages from "Carlos": "Imagen enviada" (with a small thumbnail), "aaa", "aaaa", and "Imagen enviada" (with a large black redacted image). At the bottom are "Enviar" and "Adjuntar" buttons, and a text input field "Escribe un mensaje...".

Los maestros por otra parte al ser los únicos que pueden mandar mensajes tienen el `textArea` disponible para ello, pueden enviar texto, imagen o ambos al mismo tiempo, el botón de adjuntar solo admite imágenes, además se tiene disponible el botón de “agregar alumnos” y el botón de “ver alumnos”.

Agregar alumnos a un grupo (Maestro)

The screenshot shows a search interface titled "Agregar alumnos en Estructura" with a "Atras" button. It has a search bar "Ingresa la matrícula..." and a "Buscar" button. Below is a table titled "Resultados de la búsqueda" with columns "Matrícula", "Nombre", and "Acción". One row is shown: "zs2204" and "Martin Isabel Pérez Hernández" with an "Agregar" button.

El usuario (Maestro) puede agregar un alumno buscando la matrícula proporcionada por el alumno dentro del salón de clases y dar click en “buscar” de este modo carga la tabla que aparece en la imagen dando la posibilidad de dar click en “agregar” dicho alumno o regresar a la vista anterior con el botón de atrás.

Ver alumnos de un grupo (Maestro)

The screenshot shows a user interface for managing group members. On the left, there is a sidebar with the title 'Teams UV' and three menu items: 'Inicio', 'Perfil', and 'Cerrar sesión'. The 'Cerrar sesión' item is highlighted with a blue background. The main content area has a header 'Consulta los miembros de la clase' with a 'Atras' button. Below this, a section titled 'Alumnos del Grupo' displays a table of two students. The table columns are 'Matrícula', 'Nombre', 'Correo', and 'Eliminar del grupo'. Each student row contains a red 'Eliminar' button.

Matrícula	Nombre	Correo	Eliminar del grupo
zs2282	Maria Isabel Pérez Hernández	maria.perez@example.com	Eliminar
zs2283	Gustavo Isabel Pérez Hernández	maria.perez@example.com	Eliminar

Cuando el usuario llega a a vista se despliega una lista de los alumnos inscritos a su materia, lo cual hace posible la eliminación de los mismos en el grupo dando click en el botón “Eliminar” o ir a la vista anterior (Chat) para continuar con la mensajería.

Conclusiones

A lo largo del desarrollo del proyecto, se han alcanzado varios logros clave que contribuyen a su éxito. Se logró crear una interfaz de usuario intuitiva y accesible que facilita la interacción de los usuarios, especialmente en el manejo de mensajes. La implementación del sistema de mensajería en tiempo real, utilizando RMI y sockets, permitió una comunicación eficiente entre estudiantes y maestros, garantizando que los mensajes se envíen de manera instantánea. Además, se implementó un control efectivo de roles, permitiendo que solo los maestros puedan enviar mensajes y fotos, mientras que los estudiantes solo pueden recibir y visualizar el contenido. La integración con la base de datos mediante JDBC y MySQL permitió un almacenamiento seguro y organizado de la información de los usuarios, mensajes y otros elementos esenciales para el sistema. La optimización del rendimiento, gracias al uso de Servlets y JSP.

Sin embargo, a pesar de estos logros, el proyecto presenta algunas limitaciones, aunque el sistema está preparado para manejar una cantidad moderada de usuarios, podría enfrentar dificultades al gestionar la conexión de una cantidad más grande de usuarios debido a la gestión de conexiones de sockets y RMI, que pueden consumir recursos significativos del servidor (Mi laptop) esto siendo evidente al realizar el envío de varios mensajes en corto periodo de tiempo llevando al incremento de la velocidad de los ventiladores, sin embargo esas limitaciones son técnicas del servidor y no del propio software.

Referencias

Codisi. (2022, 24 septiembre). *Crear un servidor WebSocket con Java* [Vídeo].

YouTube. <https://www.youtube.com/watch?v=fq0Ya-8M7yA>

Juan Alberto Antonio. (2020, 1 abril). *Explicación de Java RMI* [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=uamLt1pDG0M>

Jonathan Castillo. (2020, 25 marzo). *Ejemplo de RMI - Java [Actividad 4]* [Vídeo].

YouTube. <https://www.youtube.com/watch?v=xLn0THtpzhc>