**Tarea Individual 24 - Envia un correo con SMTP**

**Objetivo**

El objetivo de esta actividad es desarrollar una aplicación en Java que permita el envío de correos electrónicos utilizando el protocolo SMTP, mediante la biblioteca javax.mail. El programa deberá permitir la autenticación del usuario, la creación del mensaje con el contenido adecuado y la conexión al servidor SMTP para el envío del correo.

**Descripción de la Tarea**

Con base en el código proporcionado, se solicita a los alumnos realizar las siguientes modificaciones y desarrollos:

* **Conexión al servidor SMTP:**
  + Utilizar la clase Session de la biblioteca javax.mail para establecer la conexión con el servidor SMTP especificado.
  + Configurar las propiedades de la sesión, como el servidor SMTP y el puerto, utilizando el método setProperty.
* **Autenticación:**
  + Implementar la autenticación utilizando el usuario y la contraseña proporcionados para conectarse al servidor SMTP.
  + Mostrar un mensaje en la consola indicando si la autenticación fue exitosa o fallida.
* **Creación y envío del correo:**
  + Crear un mensaje de correo utilizando la clase MimeMessage, configurando los campos from, to, subject y body.
  + Enviar el correo electrónico utilizando el método Transport.send.
  + Mostrar en la consola si el correo fue enviado con éxito o si hubo algún error.
* **Requisitos adicionales:**
  + Gestionar las excepciones posibles (MessagingException, AuthenticationFailedException) mostrando mensajes claros en la consola.
  + Comentar el código para explicar las funcionalidades principales.
  + Asegurarse de utilizar las credenciales adecuadas y de que el servidor SMTP permita conexiones desde aplicaciones externas.
* **Uso de TLS 1.2:**
  + props.put("mail.smtp.starttls.enable", "true"); props.put("mail.smtp.ssl.protocols", "TLSv1.2");

**Entregable**

* Capturas de pantalla del código y de la ejecución del programa mostrando:

package data;

import javax.mail.\*;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import java.util.Properties;

import javax.mail.AuthenticationFailedException;

public class EnviarEmail {

public static void main(String[] args) {

// Configuración del servidor SMTP (ejemplo con Gmail)

String host = "smtp.gmail.com"; // Servidor SMTP

int port = 587; // Puerto SMTP

String user = "womens.are.women@gmail.com"; // Tu correo electrónico

String password = "ypuz bafp enyr kchb"; // Usa una clave de aplicación generada por Google

// Propiedades de la sesión

Properties props = new Properties();

props.put("mail.smtp.host", host); // Host SMTP

props.put("mail.smtp.port", String.valueOf(port)); // Puerto SMTP

props.put("mail.smtp.auth", "true"); // Autenticación requerida

props.put("mail.smtp.starttls.enable", "true"); // Habilitar TLS

props.put("mail.smtp.ssl.protocols", "TLSv1.2"); // Protocolo TLS 1.2

try {

// Crear la sesión con autenticación

Session session = Session.getInstance(props, new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(user, password);

}

});

// Deshabilitar depuración en producción

session.setDebug(false);

// Intentar autenticación

if (authenticate(session, user, password)) {

System.out.println("Autenticación exitosa.");

// Crear el mensaje de correo

Message message = createEmailMessage(

session,

user,

"heily1857@gmail.com", // Reemplaza con el destinatario real

"Prueba de Envío de Correo",

"Este es un mensaje de prueba enviado desde Java."

);

// Enviar el correo

sendEmail(message);

} else {

System.out.println("Autenticación fallida.");

}

} catch (MessagingException e) {

System.err.println("Error crítico al enviar el correo: " + e.getMessage());

e.printStackTrace(); // Mostrar detalles de la excepción para depuración

}

}

//metodo para autenticar usuario

private static boolean authenticate(Session session, String user, String password) {

Transport transport = null;

try {

transport = session.getTransport("smtp");

transport.connect(user, password);

return true;

} catch (MessagingException e) {

System.err.println("Error de autenticación: " + e.getMessage());

return false;

} finally {

if (transport != null) {

try {

transport.close();

} catch (MessagingException e) {

System.err.println("Error al cerrar el transporte: " + e.getMessage());

}

}

}

}

// Método para crear el mensaje de correo

private static MimeMessage createEmailMessage(Session session, String from, String to, String subject, String body) throws MessagingException {

MimeMessage message = new MimeMessage(session);

message.setFrom(new InternetAddress(from)); // Remitente

message.addRecipient(Message.RecipientType.TO, new InternetAddress(to)); // Destinatario

message.setSubject(subject); // Asunto

message.setText(body); // Cuerpo del mensaje

return message;

}

// Método para enviar el correo

private static void sendEmail(Message message) {

try {

Transport.send(message);

System.out.println("Correo enviado con éxito.");

} catch (MessagingException e) {

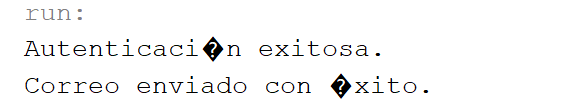
System.err.println("Error al enviar el correo: " + e.getMessage());

}

}

}

* + La conexión al servidor SMTP y la autenticación.



* + El envío del correo electrónico con éxito.

