**//Metodo para enviar correos mediante smtp con los argumentos Servidor,puerto,remitente…**

**Public class EnviadorMail {**

**public EnviadorMail**(String servidor, int puerto, String remitente, String contrasenia, String destinatario, String asunto,String cuerpo) **{**

Properties props = new Properties();

**props.put**("mail.smtp.user", remitente);

**props.put**("mail.smtp.host", servidor);

**props.put**("mail.smtp.port", puerto);

**props.put**("mail.smtp.starttls.enable", "true");

**props.put**("mail.smtp.auth", "true");

**props.put**("mail.smtp.socketFactory.port", puerto);

**props.put**("mail.smtp.socketFactory.class",

"javax.net.ssl.SSLSocketFactory");

**props.put**("mail.smtp.socketFactory.fallback", "false");

SecurityManager security = System.getSecurityManager();

**try** {

Authenticator auth = new autentificadorSMTP(remitente, contrasenia);

Session session = Session.getInstance(props, auth);

MimeMessage msg = new MimeMessage(session);

msg.setText(cuerpo);

msg.setSubject(asunto);

msg.setFrom(new InternetAddress(remitente));

msg.addRecipient(Message.RecipientType.TO, new InternetAddress(

destinatario));

Transport.send(msg);

} **catch (Exception mex)** {

}

**}**

**//clase para almacenar la estructura para conectarse y establece la Autenticacion**

**private class autentificadorSMTP extends javax.mail.Authenticator** {

**private** String remitente;

**private** String contrasenia;

**public autentificadorSMTP**(String remitente, String contrasenia) {

**this.remitente** = remitente;

**this.contrasenia** = contrasenia;

}

**public PasswordAuthentication getPasswordAuthentication**() {

return new PasswordAuthentication(remitente, contrasenia);

}

}

**public static void main(String[] args) {**

EnviadorMail EnviadorMail = new EnviadorMail("smtp.gmail.com", 465, "jarauzosinaga@gmail.com","java12xx", "hectorcarballo.hco@gmail.com", "Este es el asunto de mi correo", "Este es el cuerpo de mi correo");

}

**}**

**public class Pop3 implements Runnable {**

**private** String user;

**private** String pass;

**private** String host;

**private** PrintWriter pw;

**private** BufferedReader br;

**public Pop3(String host, String user, String pass) {**

this.user = user;

this.pass = pass;

this.host = host;

**}**

**//Método que se encarga de la lectura de correos mediante pop3 mediante Socket**

**public void run() {**

SSLSocket socket = null;

SSLSocketFactory factory =

(SSLSocketFactory) SSLSocketFactory.getDefault();

Procesar procesar = new Procesar("ayuda");

while (true) {

String correo = "";

try {

String msg = "";

socket = (SSLSocket) factory.createSocket(host, 995);

br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));

System.out.println(leerServer(br));

pw.flush();

pw.println("user " + user);

pw.flush();

System.out.println(leerServer(br));

pw.println("pass " + pass);

pw.flush();

System.out.println(leerServer(br));

System.out.print("Revisando...");

Thread.sleep(2000);

pw.println("stat");

pw.flush();

int cant = cantidad(br.readLine());

System.out.println(cant + " mensajes nuevos");

for (int i = 1; i <= cant; i++) {

pw.println("retr " + i);

pw.flush();

String data;

String subject = "";

boolean su = false;

boolean fr = false;

do {

data = br.readLine();

msg += data;

System.out.println(data);

if (data.indexOf("Subject:") != -1) {

subject = data;

su = true;

} else if (data.indexOf("From:") != -1) {

correo = data;

fr = true;

}

} while (!data.equals("."));

System.out.println("en el pop3");

try {

msg = msg.substring(msg.indexOf("<comando>"), msg.indexOf("</comando>")).substring(9).trim();

correo = correo.substring(correo.indexOf("<") + 1, correo.indexOf(">"));

System.out.println("recibido de " + correo);

subject = mensaje(subject);

procesar = new Procesar(msg);

Smtp.enviar(correo, host, procesar.procesar());

} catch (Exception e) {

Smtp.enviar(correo, "smtp.gmail.com", procesar.procesar());

}

}

for (int j = 1; j <= cant; j++) {

pw.println("dele " + j);

pw.flush();

br.readLine();

}

pw.println("quit");

pw.flush();

pw.close();

br.close();

if (socket != null) {

socket.close();

}

} catch (IOException ex) {

Logger.getLogger(Pop3.class.getName()).log(Level.SEVERE, null, ex);

} catch (InterruptedException ex) {

Logger.getLogger(Pop3.class.getName()).log(Level.SEVERE, null, ex);

}

}

**}**

**//devuelve la cantidad de correos**

**Private int cantidad(String linea) {**

String aux = linea.trim();

aux = aux.substring(aux.indexOf(" "), aux.length());

aux = aux.trim();

aux = aux.substring(0, aux.indexOf(" "));

return Integer.parseInt(aux);

**}**

**//devuelve le mensaje de un correo**

**private String mensaje(String linea) {**

String aux = linea.trim();

aux = aux.substring(aux.indexOf(":") + 1, aux.length());

aux = aux.trim();

return aux;

**}**

**//lee los datos del servidor**

**private String leerServer(BufferedReader In) throws IOException {**

String Datos = null, userInput = "";

do {

userInput = In.readLine();

if (Datos == null) {

Datos = userInput;

} else {

Datos = Datos + "\n" + userInput;

}

} while (In.ready());

return Datos;

**}**

**public static void main(String[] args) {**

Pop3 p = new Pop3("pop.gmail.com", "jarauzosinaga@gmail.com", "java12xx");

Thread t = new Thread(p);

t.start();

**}**

**}**

**public class Smtp {**

**//método para enviar correo mediante Smtp**

**public static void enviar(String correo, String host, String msgSub) {**

**try {**

new EnviadorMail(host, 465, "jarauzosinaga@gmail.com", "java12xx", correo, "Respuesta Sistema", msgSub);

**System.out.println**("Respuesta enviada a <" + correo + ">: " + msgSub);

} catch (Exception ex) {

}

**}**

**public static void main(String agrs[]) {**

**Smtp.enviar**("jarauzosinaga@gmail.com", "smtp.gmail.com", "prueba de mensaje");

**}**

**}**

**Public class Procesar {**

**private** String texto;

**public Procesar(String texto) {**

this.texto = texto.trim();

**}**

**//Método que procesa los datos del correo y ejecuta la acción correspondiente (Insertar, Listar, Modificar, Eliminar).**

**Public String procesar() {**

try {

String[] s = texto.split(" ");

if (texto.trim().toUpperCase().equals("AYUDA")) {

return ayuda();

}

if (s.length < 3) {

return "Formato no válido, para obtener ayuda, escriba el comando ayuda";

} else {

if (!s[0].trim().toUpperCase().equals("GESTIONAR")) {

return "Formato no válido, para obtener ayuda, escriba el comando ayuda";

}

String tipo = s[1].trim().toUpperCase();

if (!(tipo.equals("A") || tipo.equals("B") || tipo.equals("M") || tipo.equals("L"))) {

return "Formato no válido, para obtener ayuda, escriba el comando ayuda";

}

String tabla = s[2].trim().toUpperCase();

if (!(tabla.equals("PERSONA")

|| !tabla.equals("DOCENTE")

|| !tabla.equals("CARRERA")

|| !tabla.equals("MATERIACARRERA")

|| !tabla.equals("GESTION")

|| !tabla.equals("PREREQUISITO")

|| !tabla.equals("MATERIA"))){

return "Formato no válido, para obtener ayuda, escriba el comando ayuda";

}

if (tipo.equals("L")) {

return listar(tabla);

}

String campos = "";

for (int i = 3; i < s.length; i++) {

campos += s[i] + " ";

}

campos=campos.replace("=2C", ",");

String p[] = campos.split(",");

if (tipo.equals("B")) {

return eliminar(tabla, Integer.parseInt(p[0].trim()));

}

if (tipo.equals("A")) {

return insertar(tabla, p);

}

if (tipo.equals("M")) {

return modificar(tabla, p);

}

}

} catch (Exception ex) {

return "Formato no válido, para obtener ayuda, escriba el comando ayuda";

}

return "";

**}**

**//**

**private String ayuda() {**

return "Estos son los comandos para la Ayuda";

**}**

**//Metodo que se utiliza para listar los Datos de las Tablas**

**private String listar(String tabla) {**

if (tabla.equals("PERSONA")) {

GPersona gp = new GPersona();

try {

return "Listado de Personas\n" + gp.listarString();

} catch (Exception ex) {

return "Articulos: Ocurrio un error al listar";

}

}

if (tabla.equals("DOCENTE")) {

GDocente gd = new GDocente();

try {

return "Listado de Docente\n" + gd.listarString();

} catch (Exception ex) {

return "CLientes: Ocurrio un error al listar";

}

}

if (tabla.equals("CARRERA")) {

GCarrera gc = new GCarrera();

try {

return "Listado de carrera\n" + gc.listarString();

} catch (Exception ex) {

return "pedidos: Ocurrio un error al listar";

}

}

if (tabla.equals("MATERIACARRERA")) {

GMateriaCarrera gde = new GMateriaCarrera();

try {

return "Listado de MateriCcarrera\n" + gde.listarString();

} catch (Exception ex) {

return "Detalle: Ocurrio un error al listar";

}

}

if (tabla.equals("GESTION")) {

GGestion g = new GGestion();

try {

return "Listado de Gestion\n" + g.listarString();

} catch (Exception ex) {

return "control: Ocurrio un error al listar";

}

}

if (tabla.equals("PREREQUISITO")) {

GPreRequisito grq = new GPreRequisito();

try {

return "Listado de prerequisito\n" + grq.listarString();

} catch (Exception ex) {

return "control: Ocurrio un error al listar";

}

}

if (tabla.equals("MATERIA")) {

GMateria gmt = new GMateria();

try {

return "Listado de Materia\n" + gmt.listarString();

} catch (Exception ex) {

return "control: Ocurrio un error al listar";

}

}

return "";

**}**

**//Metodo que se Utiliza para Eliminar un dato de una Tabla**

**private String eliminar(String tabla, int id) {**

if (tabla.equals("PERSONA")){

GPersona gp = new GPersona();

try {

gp.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("DOCENTE")){

GDocente gd = new GDocente();

try {

gd.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("CARRERA")){

GCarrera gc = new GCarrera();

try {

gc.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("MATERIACARRERA")){

GMateriaCarrera gm = new GMateriaCarrera();

try {

gm.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("GESTION")){

GGestion g = new GGestion();

try {

g.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("PREREQUISITO")){

GPreRequisito grq = new GPreRequisito();

try {

grq.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

if (tabla.equals("MATERIA")){

GMateria gmt = new GMateria();

try {

gmt.eliminar(id);

} catch (Exception ex) {

return "Ocurrio un error al eliminar el registro";

}

}

return "";

**}**

**//Metodo que se Utiliza para Insertar Datos en una Tabla**

**private String insertar(String tabla, String[] campos) {**

int id = 0;

if (tabla.equals("PERSONA")) {

GPersona gp = new GPersona();

if (campos.length != 8) {

return "Parametros incorrectos";

}

try {

id = gp.insertar(campos[0].trim(), campos[1].trim(),campos[2].trim(), campos[3].trim(), campos[4].trim(), campos[5].trim(), campos[6].trim(), campos[7].trim());

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("DOCENTE")) {

GDocente gd = new GDocente();

if (campos.length != 3) {

return "Parametros incorrectos";

}

try {

id = gd.insertar(Integer.parseInt(campos[0].trim()),campos[1].trim(), Integer.parseInt(campos[2].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("CARRERA")) {

GCarrera gc = new GCarrera();

if (campos.length != 3) {

return "Parametros incorrectos";

}

try {

id = gc.insertar(campos[0].trim(),campos[1].trim(), Integer.parseInt(campos[2].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("MATERIACARRERA")) {

GMateriaCarrera gm = new GMateriaCarrera();

if (campos.length != 2) {

return "Parametros incorrectos";

}

try {

id = gm.insertar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("GESTION")) {

GGestion g = new GGestion();

if (campos.length != 3) {

return "Parametros incorrectos";

}

try {

id = g.insertar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()),Integer.parseInt(campos[2].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("PREREQUISITO")) {

GPreRequisito grq = new GPreRequisito();

if (campos.length != 2) {

return "Parametros incorrectos";

}

try {

id = grq.insertar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

if (tabla.equals("MATERIA")) {

GMateria gmt = new GMateria();

if (campos.length != 8) {

return "Parametros incorrectos";

}

try {

id = gmt.insertar(campos[0].trim(),campos[1].trim(),Integer.parseInt(campos[2].trim()),Float.parseFloat(campos[3].trim()), Float.parseFloat(campos[4].trim()),Float.parseFloat(campos[5].trim()),Float.parseFloat(campos[6].trim()),Integer.parseInt(campos[7].trim()));

} catch (Exception ex) {

return "Ocurrio un error al insertar el registro";

}

}

return ": Se inserto el registro correctamente [id=" + id + "]";

**}**

**//Metodo que se Utiliza para Modificar los Datos de una Tabla**

**Private String modificar(String tabla, String[] campos) {**

if (tabla.equals("PERSONA")) {

GPersona gp = new GPersona();

if (campos.length != 9) {

return "Parametros incorrectos";

}

try {

gp.modificar(Integer.parseInt(campos[0].trim()),campos[1].trim(), campos[2].trim(), campos[3].trim(),campos[4].trim(), campos[5].trim(), campos[6].trim(), campos[7].trim(), campos[8].trim());

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}

if (tabla.equals("DOCENTE")){

GDocente gd = new GDocente();

if (campos.length != 4) {

return "Parametros incorrectos";

}

try {

gd.modificar(Integer.parseInt(campos[0].trim()), Integer.parseInt(campos[1].trim()), campos[2].trim(),Integer.parseInt(campos[3].trim()));

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}

if (tabla.equals("CARRERA")){

GCarrera gc = new GCarrera();

if (campos.length != 4) {

return "Parametros incorrectos";

}

try {

gc.modificar(Integer.parseInt(campos[0].trim()),campos[1].trim(), campos[2].trim(),Integer.parseInt(campos[3].trim()));

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}

if (tabla.equals("MATERIACARRERA")){

GMateriaCarrera gm = new GMateriaCarrera();

if (campos.length != 3) {

return "Parametros incorrectos";

}

try {

gm.modificar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()), Integer.parseInt(campos[2].trim()));

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}

if (tabla.equals("GESTION")){

GGestion g = new GGestion();

if (campos.length != 4) {

return "Parametros incorrectos";

}

try {

g.modificar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()),Integer.parseInt(campos[2].trim()),Integer.parseInt(campos[3].trim()));

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}

if (tabla.equals("PREREQUISITO")){

GPreRequisito grq = new GPreRequisito();

if (campos.length != 3) {

return "Parametros incorrectos";

}

try {

grq.modificar(Integer.parseInt(campos[0].trim()),Integer.parseInt(campos[1].trim()), Integer.parseInt(campos[2].trim()));

} catch (Exception ex) {

return "Ocurrio un error al modificar el registro";

}

}