**PROGRAMA INGENIERÍA DE SISTEMAS**

**GUIA DE LABORATORIO**

**NOMBRE DE LA PRÁCTICA:** Laboratorio de RMI y SSL

**OBJETIVO:** implementar un método remoto que funcione de forma distribuida utilizando RMI y SSL (Security Socket Layer).

**OBJETIVOS ESPECÍFICOS:**

* Crear Interfaz de java con sus métodos
* Crear Certificados SSL para el cliente y servidor (herramienta keytool)
* Crear Clases SSL para el cliente y servidor
* Crear servidor RMI (java class implementando los métodos de la interfaz de java anteriormente creada)
* Implementar los métodos de las clases a los clientes y servidor
* Ejecutar el servidor
* Ejecutar un cliente

**TEMÁTICAS:**

**1.** InterfaceRemota: Es una interface java con todos los métodos que queramos poder invocar de forma remota,

**2.** ObjetoRemoto: Es una clase con la implementación de los métodos de InterfaceRemota**.**

**3.** Arquitectura cliente servidor

**4.** Certificados SSL

**COMPETENCIAS A DESARROLLAR:**

**1.** Implementación de la arquitectura cliente-servidor

**2.** Invocación de Métodos Remotos

**3.** Implementación de clases abstractas

**CONOCIMIENTOS PREVIOS REQUERIDOS:**

**1.** Lógica de programación

**2.** Conocimiento del lenguaje java

**RECURSOS A NIVEL HARDWARE Y SOFTWARE:**

**1.** Computador Windows

**2.** JDK java

**3.** NetBeans 11.3

**4.** Certificados SSL

**DESCRIPCIÓN DE LA PRÁCTICA (PROCEDIMIENTOS, EJERCICIOS Y METODOLOGÍA):**

1. Crear un Proyecto Java aplication en Netbeans
2. Creacion de los certificados SSL por consola usando una herramienta del jdk de java llamada keytool

* ServerPublicKey.cer
* serverKey.jks
* serverTrustedCerts.jks
* ClientPublicKey.cer
* clientKey.jks
* clientTrustedCerts.jks

1. Crear las siguientes clases java:

* CifradoDatos (java class)
* CifrarD (abstract class)
* AppletCliente(java class extends Applet)
* JFrameCliente(java class extends JFrame)
* SSLClient(java class)
* SSLServer(java class)
* DesencriptarDatos (java class)
* Server (java class implements Java interface)
* TestRemote(Java interface)

1. Programación de las clases:

* **CifradoDatos**

class CifradoDatos extends CifrarD {

public CifradoDatos(String cadena){

super(cadena);

}

public String Cifrado(){

String cadenaC="";

char c,cc;

System.out.println(cadena);

System.out.println(cadena.length());

for(int i=0;i<cadena.length();i++){

System.out.println(i);

cc=caracterCifrado(cadena.charAt(i),cadena.length(),i);

cadenaC +=cc;

System.out.print(cc);

}

return cadenaC;

}

private char caracterCifrado(char c, int lon,int i){

char cc;

int indice;

if(pc.indexOf(String.valueOf(c)) != -1){

indice=pd.indexOf(c);

cc=pc.charAt(indice);

return cc;

}

return c;

}

}

* **CifradoD**

abstract class CifrarD {

String cadena;

String pd = "ABCD EFGHIJKLMNOPQRSTUVWXYZ!-abcdefghijklmnopqrstuvwxyz0123456789";

String pc = "0Aa1\_bB2Cc3Dd4Ee5Ff6Gg7Hh8Ii !9JjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz";

public CifrarD(String cadena){

this.cadena = cadena;

}

public abstract String Cifrado();

}

* **JFrameClient**

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.logging.Level;

import java.util.logging.Logger;

import seguridad.DesencriptarDatos;

import servidor.TestRemote;

import ssl.SSLClient;

/\*\*

\*

\* @author manue

\*/

public class JFrameClient extends javax.swing.JFrame {

SSLClient sslcc ;

public JFrameClient() {

try {

sslcc= new SSLClient("localhost",5557);

} catch (Exception ex) {

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

}

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jScrollPane1 = new javax.swing.JScrollPane();

result = new javax.swing.JTextArea();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBackground(new java.awt.Color(159, 159, 213));

result.setColumns(20);

result.setRows(5);

jScrollPane1.setViewportView(result);

jButton2.setBackground(new java.awt.Color(37, 37, 156));

jButton2.setForeground(new java.awt.Color(255, 255, 255));

jButton2.setText("Saludar");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton3.setBackground(new java.awt.Color(0, 153, 0));

jButton3.setForeground(new java.awt.Color(255, 255, 255));

jButton3.setText("Palindromo");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

jLabel1.setText("CLIENTE");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addGap(25, 25, 25)

.addComponent(jScrollPane1)

.addContainerGap())

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addGap(64, 64, 64)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 112, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 113, Short.MAX\_VALUE)

.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE, 122, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(54, 54, 54))

.addGroup(layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1)

.addGap(175, 175, 175))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addGap(41, 41, 41)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 43, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE, 43, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 127, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(45, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

try {

operar("1");

} catch (RemoteException ex) {

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

} catch (NotBoundException ex) {

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

}

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

try {

operar("3");

} catch (RemoteException ex) {

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

} catch (NotBoundException ex) {

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

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(JFrameClient.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JFrameClient.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JFrameClient.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JFrameClient.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new JFrameClient().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JLabel jLabel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextArea result;

// End of variables declaration

public void operar(String opcion) throws RemoteException, NotBoundException{

Registry registry = LocateRegistry.getRegistry();

TestRemote testRemote1 = (TestRemote) registry.lookup("datoKey");

int[] arr = new int[80];

int m = 20;

for (int i = 0; i < 80; i++) {

arr[i] = m++;

}

switch (opcion) {

case "1":

System.out.println("opcion 1");

String re=testRemote1.mensajeSaludo("Manuel Cordoba");

System.out.println(re);

DesencriptarDatos desencriptarDatos = new DesencriptarDatos(re);

System.out.println(desencriptarDatos.Cifrado());

result.setText(desencriptarDatos.Cifrado());

break;

case "3":

String res=testRemote1.palindromo("ano");

result.setText(res);

break;

case "2":

break;

default:

}

TestRemote testRemote2 = (TestRemote) registry.lookup("datoKey");

}

}

* **AppletCliente**

import java.applet.Applet;

import java.awt.Button;

import java.awt.Event;

import java.awt.Graphics;

import java.awt.Label;

import java.awt.TextField;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.logging.Level;

import java.util.logging.Logger;

import seguridad.CifradoDatos;

import seguridad.DesencriptarDatos;

import servidor.TestRemote;

import ssl.SSLClient;

/\*\*

\*

\* @author manue

\*/

public class AppletCliente extends Applet {

String sal;

Label titulo = new Label("Gestion de prestamo");

Label mensaje = new Label("Ingrese salario");

Label resultado = new Label("");

TextField salario = new TextField(20);

Button button = new Button("Enviar");

SSLClient sslcc ;

public void init(){

try {

sslcc= new SSLClient("localhost",5557);

} catch (Exception ex) {

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

}

add(titulo);

add(mensaje);

add(salario);

add(button);

add(resultado);

}

public boolean action(Event event, Object ob) {

if (event.target == button) {

sal = salario.getText();

try {

operar();

} catch (RemoteException ex) {

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

} catch (NotBoundException ex) {

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

}

return true;

}

return false;

}

public void paint(Graphics g) {

}

public void operar() throws RemoteException, NotBoundException{

resultado.setText(sslcc.operar(sal));

}

}

* **DesencriptarDatos**

public class DesencriptarDatos extends CifrarD{

public DesencriptarDatos(String cadena) {

super(cadena);

}

public String Cifrado (){

String cadenac="";

char c,cc;

for(int i=0;i<cadena.length();i++){

cc=caracterDesencriptado(cadena.charAt(i),cadena.length(),i);

cadenac +=cc;

}

return cadenac;

}

private char caracterDesencriptado(char c, int lon,int i){

char cc;

int indice;

if(pd.indexOf(String.valueOf(c)) != -1){

indice=pc.indexOf(c);

cc=pd.charAt(indice);

return cc;

}

return c;

}

}

* **Server**

import java.rmi.AlreadyBoundException;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.logging.Level;

import java.util.logging.Logger;

import seguridad.CifradoDatos;

import seguridad.DesencriptarDatos;

import ssl.SSLServer;

///import javax.rmi.CORBA.Stub;

public class Server {

public static void main(String[] args) throws RemoteException, AlreadyBoundException {

try {

new SSLServer(5557).startServerWorking();

} catch (Exception ex) {

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

}

Remote ObjCliente1 = UnicastRemoteObject.exportObject(new TestRemote() {

@Override

public String mensajeSaludo(String name) throws RemoteException {

String mensaje = "Hola " + name + " BIENVENIDO, su codigo es 9900373189";

CifradoDatos cifradoDatos = new CifradoDatos(mensaje);

return cifradoDatos.Cifrado();

}

public int getData2(int[] arr) throws RemoteException {

int sum = 0;

for (int i = 0; i < 80; i++) {

sum = sum + arr[i];

}

return sum;

}

@Override

public String palindromo(String s) throws RemoteException {

s = s.replace(" ", "");

s = s.replace(",", "");

s = s.replace(".", "");

System.out.print(s);

int fin = s.length() - 1;

int ini = 0;

boolean espalin = true;

while (ini < fin) {

if (s.charAt(ini) != s.charAt(fin)) {

espalin = false;

}

ini++;

fin--;

}

if (espalin) {

return("\nEs palindromo.");

} else {

return("\nNo es palindromo.");

}

}

@Override

public String calcularPrestamo(String salario) throws RemoteException {

DesencriptarDatos desencriptarDatos = new DesencriptarDatos(salario);

int intSal = Integer.parseInt(desencriptarDatos.Cifrado());

System.out.println(intSal);

CifradoDatos cd;

if(intSal<=900000){

cd = new CifradoDatos("No ha sido aprobado su credito");

return cd.Cifrado();

}else{

cd = new CifradoDatos("Ha sido aprobado su credito");

return cd.Cifrado();

}

}

}, 0);

//Registry registry = LocateRegistry.createRegistry(Registry.REGISTRY\_PORT);

Registry registry = LocateRegistry.createRegistry(1099);

registry.bind("datoKey", ObjCliente1);

}

}

* **SSLClient**

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.security.KeyStore;

import javax.net.ssl.KeyManager;

import javax.net.ssl.KeyManagerFactory;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.TrustManager;

import javax.net.ssl.TrustManagerFactory;

import seguridad.CifradoDatos;

import seguridad.DesencriptarDatos;

import servidor.TestRemote;

public class SSLClient {

private SSLSocket client;

public SSLClient(String address, int port) throws Exception {

KeyStore keyStore = KeyStore.getInstance("JKS");

keyStore.load(new FileInputStream("src/main/certs/client/clientKey.jks"),

"clientpass".toCharArray());

KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

kmf.init(keyStore, "clientpass".toCharArray());

KeyStore trustedStore = KeyStore.getInstance("JKS");

trustedStore.load(new FileInputStream(

"src/main/certs/client/clientTrustedCerts.jks"), "clientpass"

.toCharArray());

TrustManagerFactory tmf = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());

tmf.init(trustedStore);

SSLContext sc = SSLContext.getInstance("TLS");

TrustManager[] trustManagers = tmf.getTrustManagers();

KeyManager[] keyManagers = kmf.getKeyManagers();

sc.init(keyManagers, trustManagers, null);

SSLSocketFactory ssf = sc.getSocketFactory();

client = (SSLSocket) ssf.createSocket(address, port);

client.startHandshake();

}

public void startClientWorking(){

System.out.println("Empezo Cliente");

new Thread() {

public void run() {

try {

PrintWriter output = new PrintWriter(client.getOutputStream());

output.println("Manuel");

output.flush();

System.out.println("Cliente envio");

BufferedReader input = new BufferedReader(new InputStreamReader(

client.getInputStream()));

String received = input.readLine();

System.out.println("Recivio : " + received);

client.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}.start();

}

public String operar(String sal) throws RemoteException, NotBoundException{

Registry registry = LocateRegistry.getRegistry();

TestRemote testRemote1 = (TestRemote) registry.lookup("datoKey");

CifradoDatos cifradoDatos = new CifradoDatos(sal);

String re=testRemote1.calcularPrestamo(cifradoDatos.Cifrado());

System.out.println(re);

DesencriptarDatos desencriptarDatos = new DesencriptarDatos(re);

return(desencriptarDatos.Cifrado());

}

}

* **SSLServer**

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.security.KeyStore;

import javax.net.ssl.KeyManager;

import javax.net.ssl.KeyManagerFactory;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.TrustManager;

import javax.net.ssl.TrustManagerFactory;

import seguridad.CifradoDatos;

import seguridad.DesencriptarDatos;

import servidor.TestRemote;

public class SSLClient {

private SSLSocket client;

public SSLClient(String address, int port) throws Exception {

KeyStore keyStore = KeyStore.getInstance("JKS");

keyStore.load(new FileInputStream("src/main/certs/client/clientKey.jks"),

"clientpass".toCharArray());

KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());

kmf.init(keyStore, "clientpass".toCharArray());

KeyStore trustedStore = KeyStore.getInstance("JKS");

trustedStore.load(new FileInputStream(

"src/main/certs/client/clientTrustedCerts.jks"), "clientpass"

.toCharArray());

TrustManagerFactory tmf = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());

tmf.init(trustedStore);

SSLContext sc = SSLContext.getInstance("TLS");

TrustManager[] trustManagers = tmf.getTrustManagers();

KeyManager[] keyManagers = kmf.getKeyManagers();

sc.init(keyManagers, trustManagers, null);

SSLSocketFactory ssf = sc.getSocketFactory();

client = (SSLSocket) ssf.createSocket(address, port);

client.startHandshake();

}

public void startClientWorking(){

System.out.println("Empezo Cliente");

new Thread() {

public void run() {

try {

PrintWriter output = new PrintWriter(client.getOutputStream());

output.println("Manuel");

output.flush();

System.out.println("Cliente envio");

BufferedReader input = new BufferedReader(new InputStreamReader(

client.getInputStream()));

String received = input.readLine();

System.out.println("Recivio : " + received);

client.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}.start();

}

public String operar(String sal) throws RemoteException, NotBoundException{

Registry registry = LocateRegistry.getRegistry();

TestRemote testRemote1 = (TestRemote) registry.lookup("datoKey");

CifradoDatos cifradoDatos = new CifradoDatos(sal);

String re=testRemote1.calcularPrestamo(cifradoDatos.Cifrado());

System.out.println(re);

DesencriptarDatos desencriptarDatos = new DesencriptarDatos(re);

return(desencriptarDatos.Cifrado());

}

}

* **TestRemote**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface TestRemote extends Remote {

String mensajeSaludo(String name) throws RemoteException;

public int getData2(int[] arr)throws RemoteException;

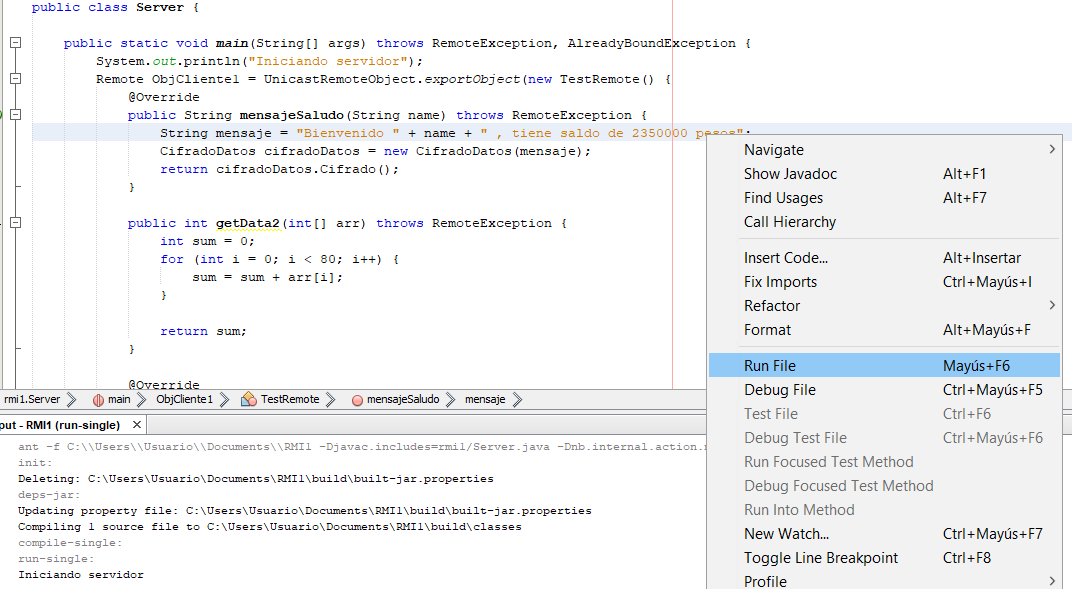
public String palindromo(String cadena)throws RemoteException;

String calcularPrestamo(String salario)throws RemoteException;

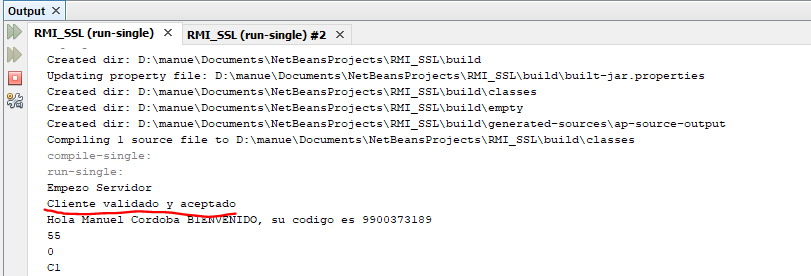
}

**PROCEDIMIENTO O DESARROLLO INGENIERÍL (Agregar imágenes de los pantallazos de funcionalidad, numerarlos y explicar cada imagen:**

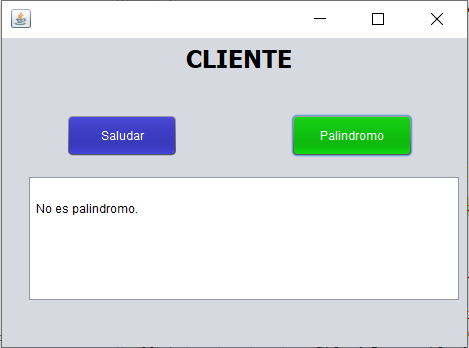
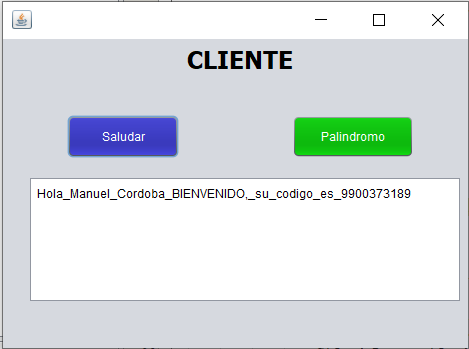
**Ejecutando server**



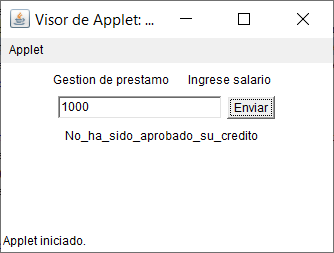
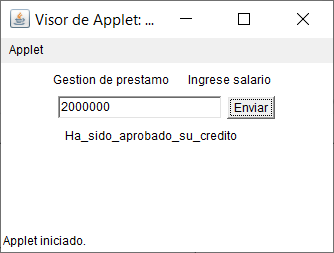
Al ejecutar un cliente



**Interfaz JFrameCliet**



**Interfaz AppletCliente**

**CRITERIOS DE EVALUACIÓN:**

1. **Se tiene en cuenta el conocimiento de las definiciones, desarrollo de la práctica, análisis de los resultados, conclusiones y las habilidades, actitud, motivación para realizar la actividad de acuerdo a la rúbrica (Ver abajo).**
2. **Terminación del laboratorio**
3. **Cumplimiento de entrega**
4. **Cumplimiento de los objetivos del laboratorio**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RÚBRICAS DE EVALUACIÓN** | | | | | |
| **CRITERIOS DE EVALUACION** | **4.5 - 5.0** | **4.0 - 4.5** | **3.0 - 3.5** | **2.5 - 3.0** | **CALIFICACION** |
| **1.- Definiciones** **Conceptos** | No es copia fiel de los textos consultados sino una síntesis de ideas completas y claras del tema. | Excede a cuatro cuartillas o no alcanza a cubrir una.  Algunos párrafos son copias fieles de los textos consultados.  Algunas ideas del tema están cortadas. | No se presenta de manera clara y completa. La relación con el problema planteado es prácticamente incongruente. El 60% del tema es copiado. | Realiza un 50% ó 60% de los experimentos, mencionando el procedimiento de manera completa incluyendo el material y equipo utilizado. Las respuestas son congruentes con los experimentos realizados. |  |
| **2.- Desarrollo de la Práctica - Dedicación** | Realiza todos los experimentos mencionando el procedimiento de manera completa, incluyendo el material y equipo utilizado. Las respuestas son congruentes con los experimentos realizados. | Realiza un 80% de los experimentos, mencionando el procedimiento de manera completa incluyendo el material y equipo utilizado. Las respuestas son congruentes con los experimentos realizados. | Realiza un 50% ó 60% de los experimentos, mencionando el procedimiento de manera completa incluyendo el material y equipo utilizado. Las respuestas son congruentes con los experimentos realizados. | Realiza un 40% de los experimentos, mencionando el procedimiento de manera completa incluyendo el material y equipo utilizado. Las respuestas son congruentes con los experimentos realizados. |  |
| **3.- Interpretación,**  **Análisis de los**  **Resultados** | Recopila y ordena los datos en relación al procedimiento. Se presentan los datos en tablas, gráficas, dibujos, etc. Claramente identificados. Los datos se interpretan y analizan comparativamente con la información bibliográfica consultada. | Presenta datos ordenados en relación al procedimiento. Se presentan en tablas, gráficas,  dibujos, etc. claramente  identificados, se interpretan y  analizan parcialmente en un  80%. | Tiene datos parcialmente ordenados, presenta algunas tablas o gráficas, los resultados  se interpretan y analizan en 50%  ó 60% | Tiene datos parcialmente ordenados, presenta algunas tablas o gráficas, los resultados se interpretan y analizan en 50%. |  |
| **4.- Conclusión** | Deduce el comportamiento de la(s) variable(s) estudiada(s) a partir del problema planteado. Rechaza o acepta la hipótesis e incluye propuestas de mejora o genera nuevos problemas. | Deduce el comportamiento de la(s) variable(s) estudiada(s) a partir del problema planteado.  Incluye el rechazo o la aceptación de la hipótesis, pero no las propuestas de mejoras. | Deduce el comportamiento de la(s) variable(s) estudiada(s) a partir del problema planteado.  No incluye el rechazo o aceptación de la hipótesis ni propone mejoras. | Deduce el comportamiento de la(s) variable(s) estudiada(s) a partir del problema planteado.  No incluye el rechazo o  aceptación de la hipótesis ni  Propone mejoras. |  |
| **5.- Actitud, Motivación, Disciplina** | Participa propositiva e integralmente en toda la práctica. | Participa ocasionalmente o lo hace constantemente pero sin coordinarse con su compañero. | Es un observador pasivo. | No participa en la realización de la práctica. |  |

**BIBLIOGRAFÍA LIBROS DIGITALES DE LA UNIVERSIDAD:**

**BIBLIOWEB:**

**AUTOR:**