 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 1 de 28

INFORME DE LABORATORIO	
ESTUDIANTES: <ul style="list-style-type: none"> Luis Felipe Velasco Tao Juan David González Dimaté 	ASIGNATURA: SISTEMAS DISTRIBUIDOS
	GRUPO: 4351
	NOTA:
CONSULTA PREVIA <ol style="list-style-type: none"> Modelo OSI: <p>El modelo de interconexión de sistemas abiertos (OSI, por sus siglas en inglés) es (Cloudflare, s.f.) un modelo conceptual, creado por la Organización Internacional de Normalización (ISO), que permite que diversos sistemas de comunicación se comuniquen usando protocolos estándar. Es decir, el modelo OSI permite que diferentes sistemas informáticos se comuniquen entre sí. Este modelo está compuesto por siete capas, las cuales son:</p> <ol style="list-style-type: none"> Aplicación: Interacción directa con los datos del usuario. Presentación: Traducción, cifrado y comprensión de los datos. Prepara los datos para el uso de la capa de aplicación. Sesión: Apertura y cierre de comunicaciones. Transporte: Comunicaciones de extremo a extremo entre dos dispositivos. Red: Posibilitar transferencias de datos entre dos redes diferentes. Enlace de datos: Posibilitar transferencias de datos en una misma red. Física: Dispositivos que participan en la transferencia de datos. Arquitectura Cliente/Servidor: Según (Fuentes, 2014) la arquitectura cliente-servidor es una estructura en donde los procesos dentro de ella toman el rol de ser clientes o servidores. En particular, los procesos de cliente interactúan con los procesos de servidor individuales en equipos anfitriones (Host) potencialmente separados, con el fin de acceder a los recursos compartidos que administran. El fin último de esta arquitectura es optimizar los recursos, de manera que todos los recursos disponibles de una organización se almacenen en un servidor, y por medio de un cliente se accedan solo a los recursos que se necesiten para un proceso o actividad en específico. Esto también permite que se puedan asegurar determinados recursos solo para usos específicos y autorizados, por medio de la asignación de privilegios a los clientes que estén utilizando el servidor. Características de la arquitectura RMI: Según (Universidad de Alicante, 2004) el RMI es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas máquinas virtuales Java, 	

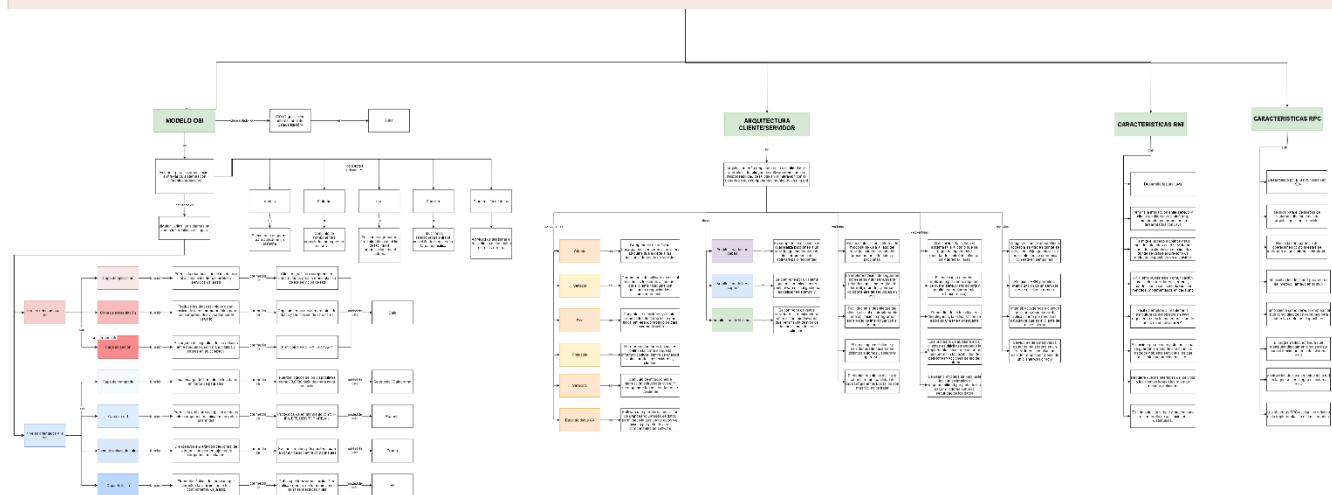
**PROGRAMA: INGENIERÍA DE SISTEMAS**


Página 2 de 28

4. Características de la arquitectura RPC: Según (Fuentes, 2014) es una variante del paradigma cliente-servidor y es una vía para implementar en la realidad este paradigma. En el RPC, un programa llama a un procedimiento localizado en otra máquina. El programador no se preocupa por las transferencias de mensajes o de las E/S. La idea de RPC es que una llamada a un procedimiento remoto se parezca lo más posible a una llamada local, esto le permite una mayor transparencia. Para lograr dicha transparencia, el RPC usa un resguardo de cliente, que se encarga de empacar los parámetros en un mensaje y le solicita núcleo que envíe el mensaje al servidor, posteriormente se bloquea hasta que regrese la respuesta.

- Manejo e implementación del mecanismo de invocación remota de métodos (RMI) en Java
- Manejo e implementación del mecanismo de llamada remota de procedimientos (RPC) en Java
- Capacidad de decisión para la implementación de RMI y RPC

Conceptos 2º Laboratorio - Sistemas Distribuidos



 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 3 de 28

RESULTADOS.

ServerSide

Interface

```
package pack;

import java.rmi.*;

public interface AdditionInterface extends Remote {

    public int add(int a, int b) throws RemoteException;

}
```

Addition

```
package pack;

import java.rmi.*;
import java.rmi.server.*;

public class Addition extends UnicastRemoteObject implements AdditionInterface {

    public Addition () throws RemoteException {    }

    public int add(int a, int b) throws RemoteException {
        return a+b;
    }

}
```

Server

```
package pack;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;


public class AdditionServer {

    public static void main(String[] argv){
        try {
            System.setSecurityManager(new SecurityManager());
            Registry registry = LocateRegistry.createRegistry(7777);
            Addition add = new Addition();
            registry.bind("ABC", add);

            // Eszte codigo esta obsoleto, por lo que no permite la posterior generación de
            // Ekeleton y Stubs debido a la versión de Java y que la herramienta RMIC esta
            // obsoleto

            //Addition Hello = new Addition();
            //Naming.bind("rmi://localhost/ABC", Hello);

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

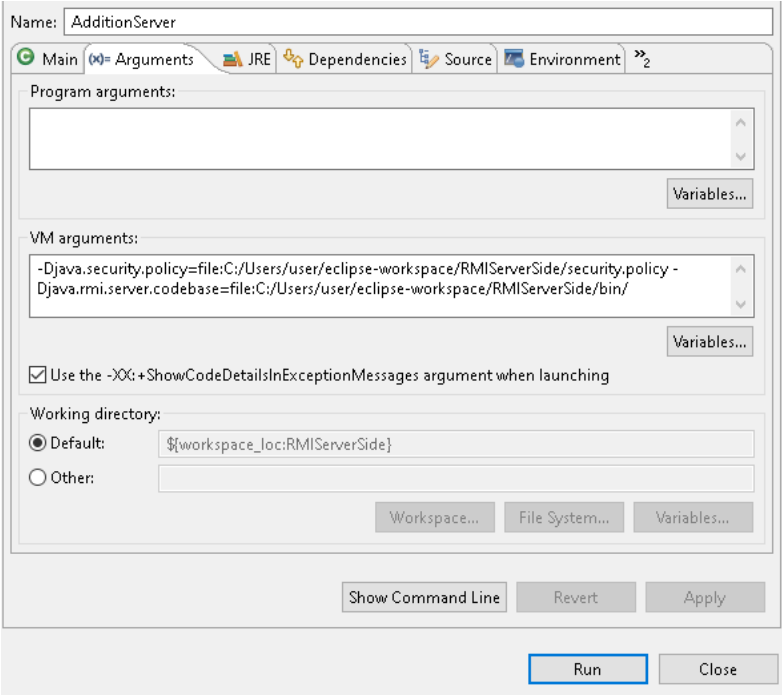
 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 4 de 28

```

        System.out.println("Addition Server is ready.");
    } catch (Exception e) {
        System.out.println("Addition Server failed: " + e);
    }
}
}

```

Argumentos



Evidencia del error con el código suministrado en el tutorial:


```

C:\Users\user\eclipse-workspace\RMIServerSide\bin\pack>rmic Addition
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
error: Invalid class file format in .\Addition.class. The major.minor version '58.0' is too recent for this tool to understand.
error: Class Addition not found.
2 errors

C:\Users\user\eclipse-workspace\RMIServerSide\bin\pack>

```

Servidor en marcha

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 5 de 28

AdditionServer [Java Application] C:\Users\user\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20210914\jre\bin\java.exe -Djava.class.path=C:\Users\user\p2\pool\workspace\AdditionServer\bin\AdditionServer.jar -Djava.class.path=C:\Users\user\p2\pool\workspace\AdditionServer\bin\AdditionServer.jar

Addition Server is ready.

ClientSide

Interface

```
package pack;

import java.rmi.*;

public interface AdditionInterface extends Remote {

    public int add(int a, int b) throws RemoteException;

}
```

Argumentos

Name: AdditionClient

Main Arguments: JRE Dependencies Source Environment Common

Program arguments:

Variables...

VM arguments:

-Djava.security.policy=file:C:/Users/user/eclipse-workspace/RMIClientSide/security.policy -Djava.rmi.server.codebase=file:C:/Users/user/eclipse-workspace/RMIClientSide/bin/

Variables...

☒ Use the -XOg+ShowCodeDetailsInExceptionMessages argument when launching

Working directory:

☒ Default: \${workspace_loc:RMIClientSide}

☐ Other:


Workspace... File System... Variables...

Show Command Line Revert Apply

Run Close

Cliente

```
package pack;
```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 6 de 28

```
import java.rmi.*;

public class AdditionClient {
    public static void main(String[] args) {


        AdditionInterface hello;
        try {
            System.setSecurityManager(new SecurityManager());
            hello = (AdditionInterface) Naming.lookup("rmi://localhost:7777/ABC");
            //Codigo obsoleteto
            //hello = (AdditionInterface) Naming.lookup("rmi://localhost/ABC");
            int result = hello.add(9, 10);
            System.out.println("Result is :" + result);
        } catch (Exception e) {
            System.out.println("HelloClient exception: " + e);
        }
    }
}
```

Puesta en marcha cliente

<terminated> AdditionClient [Java Application] C:\Users\user\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.jre\bin\java.exe
Result is :19

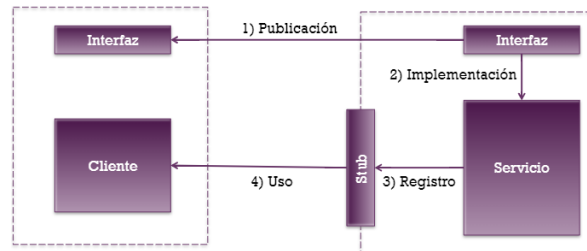
Estructura de los proyectos

- ✓ RMIClientSide
 - > JRE System Library [JavaSE-14]
 - ✓ src
 - ✓ pack
 - > AdditionClient.java
 - > AdditionInterface.java
 - security.policy
- ✓ RMIServerSide
 - ✓ src
 - ✓ pack
 - > Addition.java
 - > AdditionInterface.java
 - > AdditionServer.java
 - > JRE System Library [JavaSE-14]
 - security.policy

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 7 de 28

CUESTIONARIO.

1. Crea e implementa una Interfaz gráfica, servidor y un cliente, según la arquitectura RMI, tenga en cuenta la siguiente imagen.



Tomado de: <https://docplayer.es/2060761-Java-rmi-sistemas-distribuidos-rodrigo-santamaria.html>

Respuesta:

Servidor

```

package interfacermi;

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

public interface InterfaceJava extends Remote{

    public String getData(String name) throws RemoteException;
    public String getSuma(String arr) throws RemoteException;
}
  
```


```

package server;

import interfacermi.InterfaceJava;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.AlreadyBoundException;
import lib.CifradoDato;

public class RmiServer {

    public static void main(String[] args) throws RemoteException, AlreadyBoundException {
        CifradoDato cd = new CifradoDato();
        System.out.println(" - INICIANDO SERVIDOR - ");
        Remote skeleton = UnicastRemoteObject.exportObject(new InterfaceJava() {
            @Override
            public String getData(String name) throws RemoteException {
                System.out.println("PETICION DATA");
                return cd.getCifrado("Retorno desde el servidor = " + cd.getDescifrado(name) + "");
            }
            @Override
            public String getSuma(String arr) throws RemoteException {
                int x = 0;
                String[] ar = cd.getDescifrado(arr).split(",");
                System.out.println("PETICION SUMA");
                for (String j : ar) {
                    x += Integer.parseInt(j);
                }
            }
        });
    }
}
  
```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 8 de 28

```

        return cd.getCifrado(String.valueOf(x));
    }

    }, 0);

    Registry registry = LocateRegistry.createRegistry(7777);
    registry.bind("ABC", skeleton );
}
}
}

```

Cliente

```

package interfacermi;

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

public interface InterfaceJava extends Remote{

    public String getData(String name) throws RemoteException;
    public String getSuma(String arr) throws RemoteException;
}

```

```

package client;

import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import interfacermi.InterfaceJava;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import lib.CifradoDato;

/**
 *
 * @author user
 */
public class Cliente1G extends javax.swing.JFrame {


    InterfaceJava obj;
    CifradoDato cd = new CifradoDato();

    public Cliente1G() throws RemoteException, NotBoundException, MalformedURLException {
        this.obj = (InterfaceJava) Naming.lookup("rmi://192.168.0.14:7777/ABC");
        setTitle("CLIENTE RMI");
        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() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextFieldData = new javax.swing.JTextField();
    }
}

```


 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO		
	PROGRAMA: INGENIERÍA DE SISTEMAS		
	LAB-02	Versión: 1	Fecha: 24/09/2021
			Página 9 de 28

```

jButtonGetDAta = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();
jTextFieldNumB = new javax.swing.JTextField();
jButtonSumar = new javax.swing.JButton();
jTextFieldNumA = new javax.swing.JTextField();
jLabelGetData = new javax.swing.JLabel();
jLabelSuma = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(188, 199, 239));

jLabel11.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N
jLabel11.setText("CLIENTE RMI");

jLabel2.setFont(new java.awt.Font("Arial", 1, 11)); // NOI18N
jLabel2.setText("OBTENER INFORMACIÓN");

jTextFieldData.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
jTextFieldData.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextFieldDataActionPerformed(evt);
    }
});

jButtonGetDAta.setBackground(new java.awt.Color(153, 204, 255));
jButtonGetDAta.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
jButtonGetDAta.setText("Obtener");
jButtonGetDAta.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonGetDAtaActionPerformed(evt);
    }
});

jLabel3.setFont(new java.awt.Font("Arial", 1, 11)); // NOI18N
jLabel3.setText("SUMAR VALORES ENTEROS");

jTextFieldNumB.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N


jButtonSumar.setBackground(new java.awt.Color(255, 153, 153));
jButtonSumar.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
jButtonSumar.setText("Sumar");
jButtonSumar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonSumarActionPerformed(evt);
    }
});

jTextFieldNumA.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
jTextFieldNumA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextFieldNumAActionPerformed(evt);
    }
});

jLabelGetData.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
jLabelSuma.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(63, 63, 63)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel3)
                .addGroup(layout.createSequentialGroup()
                    .addGap(63, 63, 63)
                    .addComponent(jTextFieldNumA)
                    .addComponent(jTextFieldNumB)
                    .addComponent(jButtonSumar)
                    .addComponent(jButtonGetDAta)
                    .addComponent(jLabelGetData)
                    .addComponent(jLabelSuma)
                    .addComponent(jLabel11)
                    .addComponent(jLabel2)
                    .addComponent(jTextFieldData)
                )
            )
        )
    );

```


 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 10 de 28

```

        .addComponent(jLabel12)
        .addGap(105, 105, 105)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabelGetData, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
            .addComponent(jLabelSuma, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jTextFieldNumA, javax.swing.GroupLayout.PREFERRED_SIZE, 283,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldData, javax.swing.GroupLayout.PREFERRED_SIZE, 283,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextFieldNumB, javax.swing.GroupLayout.PREFERRED_SIZE, 283,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(0, 0, Short.MAX_VALUE))))
        .addGap(76, 76, 76)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jButtonGetData, javax.swing.GroupLayout.PREFERRED_SIZE, 153,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButtonSumar, javax.swing.GroupLayout.PREFERRED_SIZE, 153,
                javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(21, 21, 21))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel11)
            .addGap(320, 320, 320))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(22, 22, 22)
                .addComponent(jLabel11)
                .addGap(43, 43, 43)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel12)
                    .addComponent(jTextFieldData, javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButtonGetData))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabelGetData, javax.swing.GroupLayout.PREFERRED_SIZE, 21,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(13, 13, 13)
                        .addComponent(jTextFieldNumA, javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(72, 72, 72)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel13)
                            .addComponent(jButtonSumar))))
                .addGap(2, 2, 2)
                .addComponent(jTextFieldNumB, javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jLabelSuma, javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE)
                .addContainerGap())
            );
    pack();
}

private void jTextFieldNumAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 11 de 28

```

private void jButtonSumarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int a = Integer.parseInt(jTextFieldNumA.getText());
        int b = Integer.parseInt(jTextFieldNumB.getText());
        String arr = a+", "+b;
        int r = Integer.parseInt(cd.getDescifrado(obj.getSuma(cd.getCifrado(arr))));
        jLabelSuma.setText(a+ " + " +b+ " = "+r);
        jTextFieldNumA.setText("");
        jTextFieldNumB.setText("");
    } catch (NumberFormatException e){
        JOptionPane.showMessageDialog(null, "Solo se admiten valores numericos enteros");
        jTextFieldNumA.setText("");
        jTextFieldNumB.setText("");
    } catch (RemoteException ex) {
        Logger.getLogger(Cliente1G.class.getName()).log(Level.SEVERE, null, ex);
    }
}


private void jTextFieldDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButtonongETdATAActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String nombre = cd.getDescifrado(obj.getData(cd.getCifrado(jTextFieldData.getText())));
        jTextFieldData.setText("");
        jLabelGetData.setText(nombre);
    } catch (RemoteException ex) {
        Logger.getLogger(Cliente1G.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(Cliente1G.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Cliente1G.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Cliente1G.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Cliente1G.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    }
    </editor-fold>
    </editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {

```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 12 de 28

```

@Override
public void run() {
    try {
        new Cliente1G().setVisible(true);
    } catch (RemoteException | NotBoundException ex) {
        Logger.getLogger(Cliente1G.class.getName()).log(Level.SEVERE, null, ex);
    } catch (MalformedURLException ex) {
        Logger.getLogger(Cliente1G.class.getName()).log(Level.SEVERE, null, ex);
    }
}
});
}

// Variables declaration - do not modify
private javax.swing.JButton jButtonSumar;
private javax.swing.JButton jButtonongETdATA;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabelGetData;
private javax.swing.JLabel jLabelSuma;
private javax.swing.JTextField jTextFieldData;
private javax.swing.JTextField jTextFieldNumA;
private javax.swing.JTextField jTextFieldNumB;
// End of variables declaration
}

```

Cifrado de datos:

Para el cifrado de datos se creó una librería con el siguiente contenido, de la cual se creó posteriormente el archivo.jar que se incluyó en esta actividad:

```

package lib;

abstract class cifradoD {

    String P1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789",
        P2 = "Aa0Bb1Cc2Dd3Ee4Ff5Gg6Hh7Ii8Jj9KkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz";
    int L = P1.length();
    int X = P2.length();

    public abstract String getCifrado(String cadenac);


    public abstract String getDescifrado(String cadenac);
}

public class CifradoDato extends cifradoD {

    @Override
    public String getCifrado(String cadena) {
        String cc = "";
        int l = cadena.length();
        for (int i = 0; i < l; i++) {
            char x = cadena.charAt(i);
            cc += getCaracter(x, l, i, 'C');
        }
        return cc;
    }

    private char getCaracter(char c, int l, int p, char m) {
        if (P1.indexOf(c) == -1) {
            return c;
        } else {
            int pp = 0;
            switch (m) {
                case 'C':

```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 13 de 28

```

        pp = P1.indexOf(c) + 1 + p;
        if (pp - L >= 0) {
            pp -= L;
        }
        return P2.charAt(pp);
    case 'D':
        pp = P2.indexOf(c) - 1 - p;
        if (pp < 0) {
            pp += L;
        }
        return P1.charAt(pp);
    default:
        return c;
    }
}
}

@Override
public String getDescifrado(String cadenac) {
    String cc = "";
    for (int i = 0; i < cadenac.length(); i++) {
        cc += getCaracter(cadenac.charAt(i), cadenac.length(), i, 'D');
    }
    return cc;
}
}

```

Funcionamiento



The screenshot shows a Java Swing window titled "CLIENTE RMI". The window has a light gray background and contains two main sections:

- OBTENER INFORMACIÓN:** This section has a text input field, a blue button labeled "Obtener", and a label that displays "Retorno desde el servidor = Felipe".
- SUMAR VALORES ENTEROS:** This section has two text input fields, a red button labeled "Sumar", and a label that displays the result "44 + 50 = 94".

Below the window, there is an "Output" console window. It shows two tabs: "Servidor_RMI (run)" and "Cliente_RMI (run)". The "Cliente_RMI (run)" tab is active and displays the following output:

```

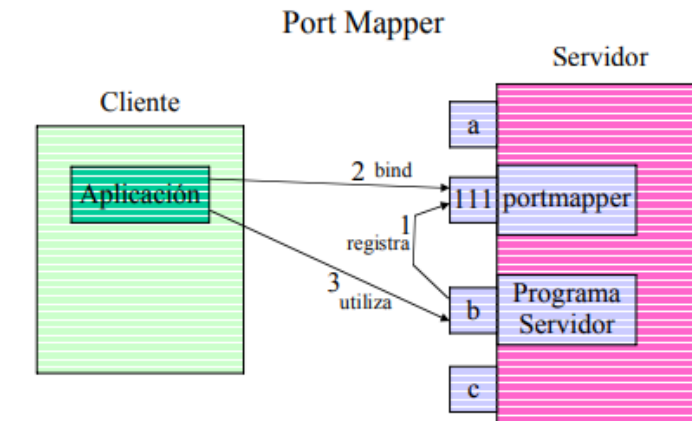
run:
- INICIANDO SERVIDOR -
PETICION DATA
PETICION SUMA

```

2. Observa la siguiente imagen y realiza la debida implementación del proceso RPC



Servicios sobre RPC



Respuesta

- De lado del servidor:

Creamos una interfaz que establece el endpoint y los servicios que se van a utilizar en el programa.

```
package rpc_ejemplo;

import javax.ws.WebMethod;
import javax.ws.WebService;
import javax.ws.soap.SOAPBinding;
import javax.ws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style = Style.RPC)
public interface InterfazServicios {
    @WebMethod String getMessage(String name);
}
```

Luego creamos una clase que implementará este endpoint junto a sus servicios disponibles.


```
package rpc_ejemplo;

import javax.ws.WebService;

@WebService(endpointInterface = "rpc_helloworld.InterfazServicios")
public class ImplServicios implements InterfazServicios{

    @Override
    public String getMessage(String name) {
        return name;
    }
}
```

Finalmente creamos un Publisher que va a desplegar el servicio web y crea y publica el endpoint por el objeto implementador específico en una dirección dada, que en este caso será <http://localhost:7779/ws/servicio>.

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 15 de 28

- **De lado del cliente:**

Creamos una clase que va a consumir el servicio web previamente establecido.

```
import javax.xml.namespace.QName;
import javax.xml.ws.Service;

public class RPC_Mensaje {

    public static void main(String[] args) {

        try {
            //Refer to wsdl document
            URL url = new URL("http://localhost:7779/ws/servicio?wsdl");


            //Refer to wsdl document
            QName qname = new QName("http://rpc_ejemplo/", "ImplServiciosService");

            Service service = Service.create(url, qname);
            InterfazServicios mensaje = service.getPort(InterfazServicios.class);

            System.out.println(mensaje.getMessage("Utilizando Servicio!"));

        } catch (MalformedURLException ex) {
            System.out.println("WSDL document url error: " + ex);
        }
    }
}
```

Luego de haber realizado lo anterior, ejecutamos el Publisher y luego ingresamos a la URL que le habíamos establecido.

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 16 de 28

< > ↺ VPN
localhost:7779/ws/servicio?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f
▼ <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/soap/"
xmlns:tns="http://rpc_ejemplo/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://rpc_ejemplo/" name="ImplServiciosService">
  <types/>
  ▼ <message name="getMessage">
    <part name="arg0" type="xsd:string"/>
  </message>
  ▼ <message name="getMessageResponse">
    <part name="return" type="xsd:string"/>
  </message>
  ▼ <portType name="InterfazServicios">
    ▼ <operation name="getMessage">
      <input wsam:Action="http://rpc_ejemplo/InterfazServicios/getMessageRequest" message="tns:getMessage"/>
      <output wsam:Action="http://rpc_ejemplo/InterfazServicios/getMessageResponse" message="tns:getMessageResponse"/>
    </operation>
  </portType>
  ▼ <binding name="ImplServiciosPortBinding" type="tns:InterfazServicios">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    ▼ <operation name="getMessage">
      <soap:operation soapAction=""/>
      ▼ <input>
        <soap:body use="literal" namespace="http://rpc_ejemplo/">
      </input>
      ▼ <output>
        <soap:body use="literal" namespace="http://rpc_ejemplo/">
      </output>
    </operation>
  </binding>
  ▼ <service name="ImplServiciosService">
    ▼ <port name="ImplServiciosPort" binding="tns:ImplServiciosPortBinding">
      <soap:address location="http://localhost:7779/ws/servicio"/>
    </port>
  </service>
</definitions>

```

Se debe copiar y pegar los datos que se encuentran en targetNameSpace y name a los parámetros del QName en la clase de Cliente.

Para finalizar ejecutamos el programa, que debe lanzar el mensaje que le establecimos en el lado del cliente.



UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021

Página 17 de
28

The screenshot shows an IDE with several tabs: Start Page, InterfazServicios.java, ImplServicios.java, Publisher.java, and RPC_Mensaje.java. The active tab is RPC_Mensaje.java, displaying the following Java code:

```
7 import javax.xml.namespace.QName;
8 import javax.xml.ws.Service;
9
10 public class RPC_Mensaje {
11
12     public static void main(String[] args) {
13
14         try {
15             //Refer to wsdl document
16             URL url = new URL("http://localhost:7779/ws/servicio?wsdl");
17
18             //Refer to wsdl document
19             QName qname = new QName("http://rpc_ejemplo/", "ImplServiciosService");
20
21             Service service = Service.create(url, qname);
22             InterfazServicios mensaje = service.getPort(InterfazServicios.class);
23
24             System.out.println(mensaje.getMessage("Utilizando Servicio!"));
25
26         } catch (MalformedURLException ex) {
27             System.out.println("WSDL document url error: " + ex);
28         }
29     }
30 }
31
```

Below the code editor, the Output window shows the following text:

```
run:
Utilizando Servicio!
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Crea un proceso RPC en java que permita interactúa con un servidor y un cliente, para que realice el siguiente proceso. El servidor envía 2 números y el cliente hace la suma y devuelve el resultado.

Respuesta

Creamos el endpoint del Web service.



UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021

Página **18** de
28

```
package com.edu.usbbog.rpc_suma;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

/**
 *
 * @author Dave
 */
@WebService
@SOAPBinding(style = SOAPBinding.Style.RPC)
public interface InterfazServicios {
    @WebMethod String getSuma(int a, int b);
}
```

Creamos la implementación del Web Service.


```
/**
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.edu.usbbog.rpc_suma;

import javax.jws.WebService;

@WebService(endpointInterface = "com.edu.usbbog.rpc_suma.InterfazServicios")
public class ImplSuma implements InterfazServicios{

    @Override
    public String getSuma(int a, int b) {
        int aux = a+b;
        return a+" + "+b + " = " + String.valueOf(aux);
    }
}
```

Creamos el Publisher.

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 19 de 28

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.edu.usbbog.rpc_suma;

import javax.xml.ws.Endpoint;

/**
 *
 * @author Dave
 */
public class Publisher {
    public static void main(String[] args) {
        Endpoint.publish("http://localhost:7779/ws/servicio", new ImplSuma());
    }
}

```

Finalmente creamos el Cliente.



UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021

Página **20** de
28

```
import javax.swing.JOptionPane;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;

/**
 *
 * @author Dave
 */
public class RPC_suma {

    public static void main(String[] args) {
        try {
            //Refer to wsdl document
            URL url = new URL("http://localhost:7779/ws/servicio?wsdl");

            //Refer to wsdl document
            QName qname = new QName("http://rpc_suma.usbbog.edu.com/", "ImplSumaService");

            Service service = Service.create(url, qname);
            InterfazServicios suma = service.getPort(InterfazServicios.class);
            String aux = JOptionPane.showInputDialog("Ingrese un número");
            int a = Integer.parseInt(aux);
            aux = JOptionPane.showInputDialog("Ingrese otro número");
            int b = Integer.parseInt(aux);
            System.out.println(suma.getSuma(a, b));

        } catch (MalformedURLException ex) {
            System.out.println("WSDL document url error: " + ex);
        }
    }
}
```

Ahora ejecutamos el Publisher, obtenemos los datos para el QName y ejecutamos el programa.



UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021

Página **21** de
28

The screenshot shows an IDE with the following components:

- Projects:** A tree view on the left showing the project structure, including `RPC_Suma` and its sub-packages.
- Source:** The main editor window displaying the `RPC_Suma.java` file. The code includes imports for `javax.swing.JOptionPane`, `javax.xml.namespace.QName`, and `javax.xml.ws.Service`. The `main` method creates a `Service` object, gets a `Port` from `InterfazServicios.class`, and uses `JOptionPane` to prompt the user for two numbers. The result is printed using `System.out.println`.
- Output:** A console window at the bottom showing the output of the program. It displays the command `exec-maven-plugin:1.5.0:exec (default-cli) @ RPC_Suma ---` and the result `4 + 6 = 10`. The output also indicates `BUILD SUCCESS` and provides the total time and finish time.

4. Crea un proceso que unifique las arquitecturas RMI Y RPC y emita un mensaje “CLIENTE SERVIDOR”.

Respuesta

Para fusionar las dos arquitecturas, iniciamos con la interfaz en la cual se usarán los elementos requeridos para la conexión RMI y RPC, esto por medio de los imports para RPC, sus notaciones y la delimitación para conexiones RMI

The screenshot shows the `InterfazServicios.java` file in an IDE. The code defines a package `rpc_ejemplo` and imports the following classes:

- `javax.jws.WebMethod`
- `javax.jws.WebService`
- `javax.jws.soap.SOAPBinding`
- `javax.jws.soap.SOAPBinding.Style`
- `java.rmi.Remote`
- `java.rmi.RemoteException`

The interface is annotated with `@WebService` and `@SOAPBinding(style = Style.RPC)`. It defines a method `getMessage` that takes a `String` message and returns a `String`, with a `throws RemoteException` declaration.



UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021


Página **22** de
28

Luego de interfaz en la cual también se define el único método a emplear en este ejercicio, se implementará esta interfaz a una clase en donde se defina el comportamiento del método y se asegure que la clase responderá a RMI y RPC.

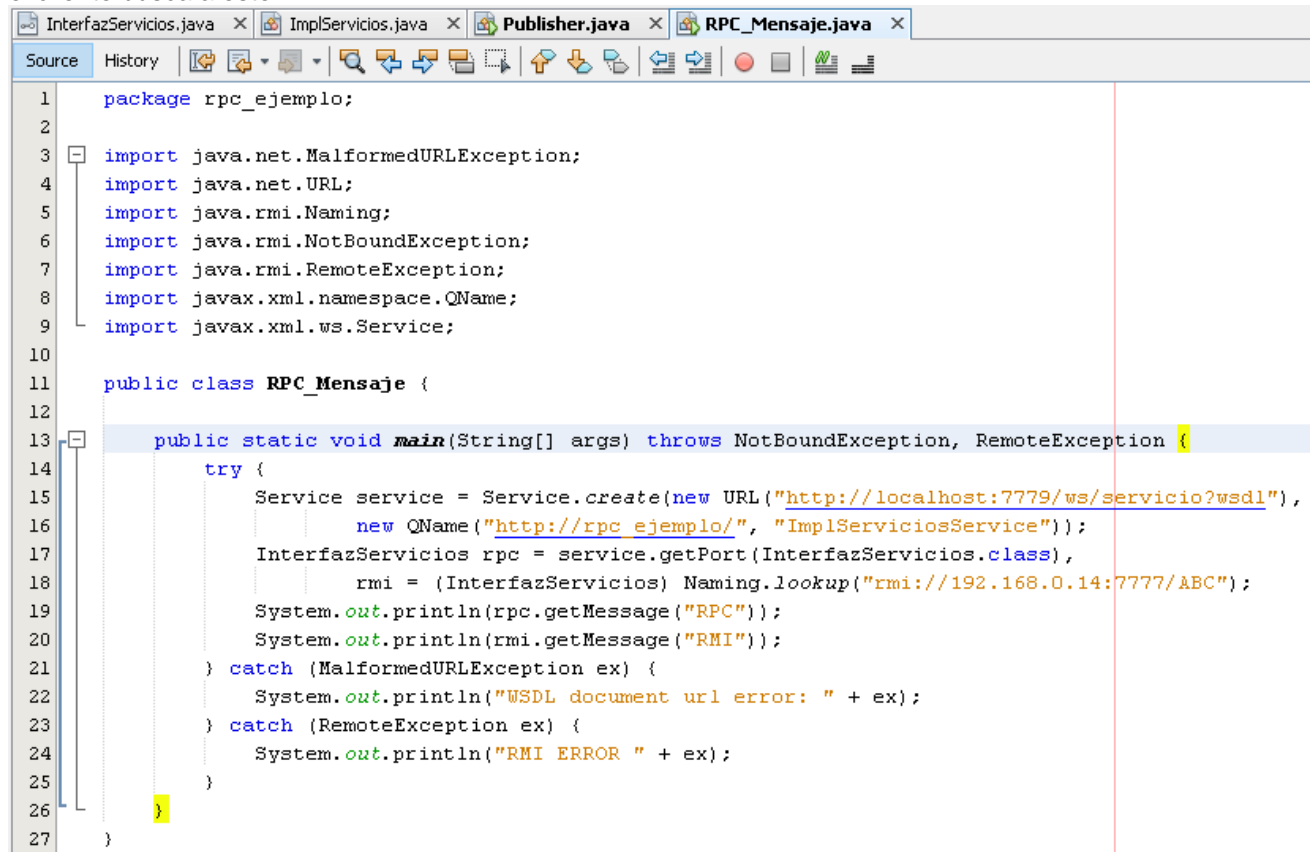
```
InterfazServicios.java x ImplServicios.java x
Source History
1 package rpc_ejemplo;
2
3 import javax.jws.WebService;
4 import java.rmi.RemoteException;
5 import java.rmi.server.UnicastRemoteObject;
6
7 @WebService(endpointInterface = "rpc_ejemplo.InterfazServicios")
8 public class ImplServicios extends UnicastRemoteObject implements InterfazServicios {
9
10     public ImplServicios() throws RemoteException { }
11
12     @Override
13     public String getMessage(String mensaje) throws RemoteException {
14         System.out.println(mensaje + " SOLICITUD");
15         return "Cliente servidor " + mensaje;
16     }
17 }
```

Con la clase definida e implementado la interfaz en esta, se define el publicador RPC / servidor RMI en el cual se invoquen los elementos para cada una de las arquitecturas:

```
InterfazServicios.java x ImplServicios.java x Publisher.java x
Source History
1 package rpc_ejemplo;
2
3 import java.rmi.AlreadyBoundException;
4 import java.rmi.RemoteException;
5 import java.rmi.registry.LocateRegistry;
6 import java.rmi.server.RemoteObject;
7 import javax.xml.ws.Endpoint;
8
9 public class Publisher {
10
11     public static void main(String[] args) throws RemoteException,
12         AlreadyBoundException {
13         System.out.println(" - INICIANDO SERVIDOR - ");
14         Endpoint.publish("http://localhost:7779/ws/servicio",
15             new ImplServicios());
16         LocateRegistry.createRegistry(7777).bind("ABC",
17             RemoteObject.toStub(new ImplServicios()));
18     }
19 }
```

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 23 de 28

Finalmente, se define al cliente el cual debe ser capaz de trabajar con los paths definidos para cada una de las arquitecturas, las dos arquitecturas, tanto en su cliente como en su servidor, no pueden ser completamente fusionadas ya que una trabaja puramente sobre RMI y la otra emplea servicios web, por lo que, los únicos elementos que no se pueden unir son la forma en que se abre el servidor y la forma en que el cliente busca a este:

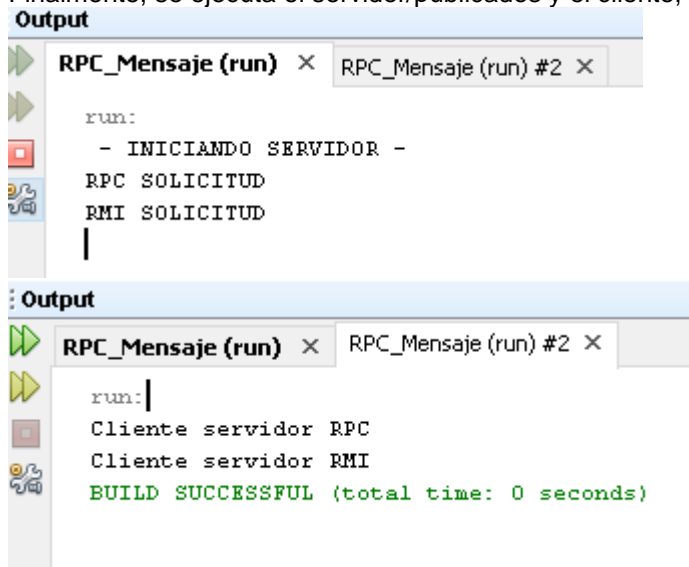


```

1  package rpc_ejemplo;
2
3  import java.net.MalformedURLException;
4  import java.net.URL;
5  import java.rmi.Naming;
6  import java.rmi.NotBoundException;
7  import java.rmi.RemoteException;
8  import javax.xml.namespace.QName;
9  import javax.xml.ws.Service;
10
11 public class RPC_Mensaje {
12
13     public static void main(String[] args) throws NotBoundException, RemoteException {
14         try {
15             Service service = Service.create(new URL("http://localhost:7779/ws/servicio?wsdl"),
16                 new QName("http://rpc_ejemplo/", "ImplServiciosService"));
17             InterfazServicios rpc = service.getPort(InterfazServicios.class),
18                 rmi = (InterfazServicios) Naming.lookup("rmi://192.168.0.14:7777/ABC");
19             System.out.println(rpc.getMessage("RPC"));
20             System.out.println(rmi.getMessage("RMI"));
21         } catch (MalformedURLException ex) {
22             System.out.println("WSDL document url error: " + ex);
23         } catch (RemoteException ex) {
24             System.out.println("RMI ERROR " + ex);
25         }
26     }
27 }

```

Finalmente, se ejecuta el servidor/publicados y el cliente, y se obtiene el siguiente resultado:



```


Output
RPC_Mensaje (run) × RPC_Mensaje (run) #2 ×

run:
- INICIANDO SERVIDOR -
RPC SOLICITUD
RMI SOLICITUD
|

Output
RPC_Mensaje (run) × RPC_Mensaje (run) #2 ×

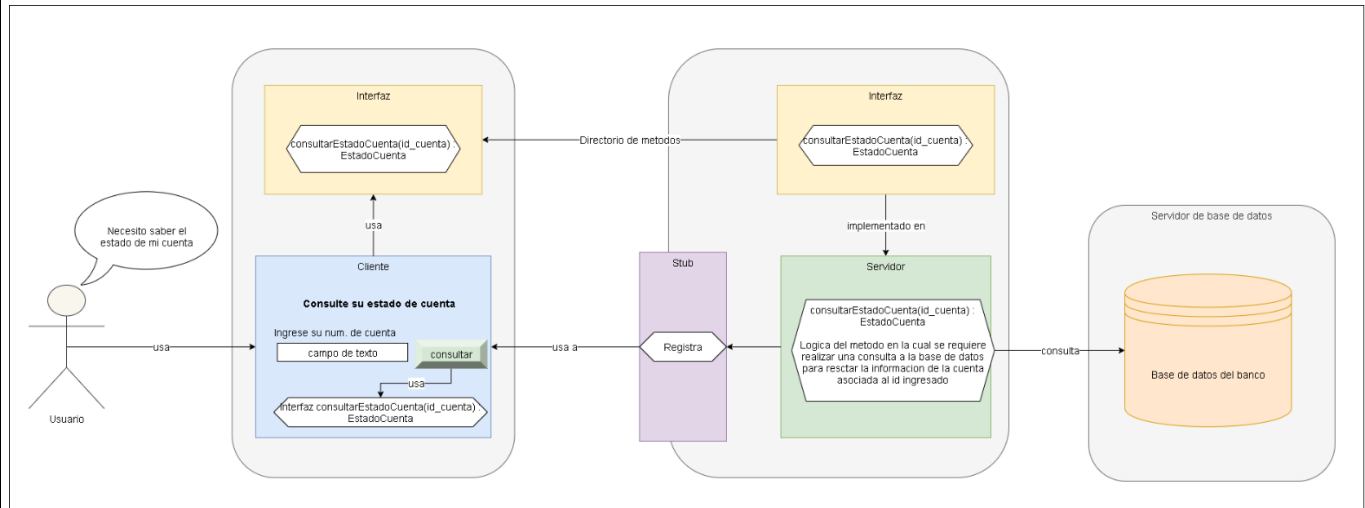
run:
Cliente servidor RPC
Cliente servidor RMI
BUILD SUCCESSFUL (total time: 0 seconds)

```

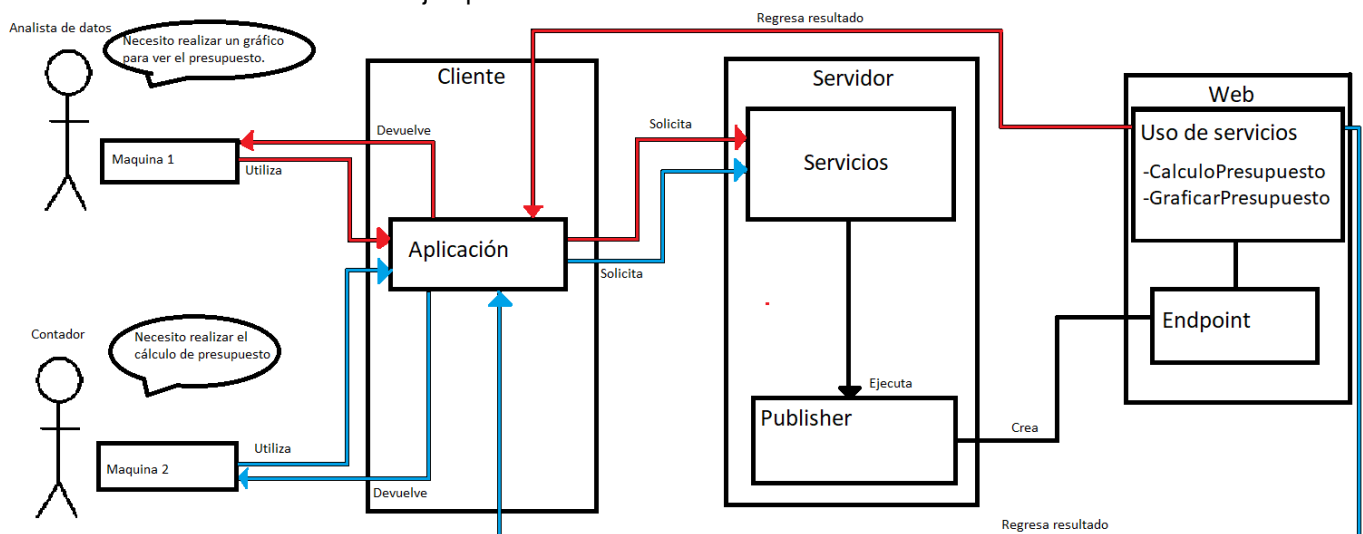
 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 24 de 28

5. Explique mediante un gráfico de elaboración propia como función la arquitectura RMI. La explicación debe estar situada en un ejemplo en el contexto real.

Respuesta



6. Explique mediante un gráfico de elaboración propia como función la arquitectura RPC. La explicación debe estar situada en un ejemplo en el contexto real.




Respuesta

7. Cita y explica las características que hacen partes de los RPC.

Respuesta

Según (Amazon Web Services, 2021; How, 2020; Perez, 2015), se pueden definir las siguientes características para RPC en los sistemas distribuidos:

- El protocolo RPC fue desarrollado por Andrew Birrell y Bruce Nelson en 1984 como un mecanismo que les permitiera de forma síncrona dentro de una red el flujo de datos y control para la ejecución de procedimientos en un servidor encargado de retornar el resultado de cada procedimiento al respectivo cliente que realizó la llamada.

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 25 de 28

- RPC está orientado a sistemas distribuidos con arquitectura Cliente – servidor o cualquier modelo arquitectónico el cual defina a un servidor en trabajo de realizar las tareas de procesamiento de datos y peticiones realizadas por otro equipo.
- Comúnmente se emplea para la comunicación en redes punto a punto, de modo tal se asegure fácilmente la conexión directa entre el cliente y el servidor.
- La operación de RPC se enfoca en el bloqueo del servicio y el cliente hasta que este último reciba una respuesta a la petición o llamado que realizo al servidor.
- Sin que el servidor RPC se encuentre activo, los nodos cliente no puede encontrarse activos.
- Para la comunicación entre el cliente y servidor se requiere de un Stub a cada lado el cual permita la comunicación y registro de las direcciones IP, además de la correlación de los procesos que pueden ser invocados.

CAUSAS DE ERROR Y ACCIONES PARA OBTENER MEJORES RESULTADOS.


- El despliegue de RMI por medio de RMIC ha quedado obsoleta, además que la implementación de políticas de seguridad por medio de Eclipse es un poco demorada.
- El despliegue de un cliente RMI con interfaz gráfica en Kali Linux no fue posible, aun cuando se configuro la IP del servidor y el puerto.
- La no puesta en marcha del publicador, o que este tenga un fallo en su ejecución, generara que todos los clientes que estén usando sus servicios no puedan realizar sus actividades correctamente.
- La fusión de las arquitecturas RMI y RPC debe contemplar la implementación en las interfaces, clientes y servidores todos los elementos requeridos para el uso de ambas arquitecturas, sin olvidar que cada una emplea características de comunicación diferentes, por lo que se tendrían en realidad dos canales de comunicación entre el cliente y el servidor.

CONCLUSIONES.

- El despliegue de sistemas en donde se implementa el protocolo RMI es una solución que puede ayudar en redes LAN que requieren el acceso a procesos e información para las actividades de alguna organización
- Se requiere consultar la documentación de RMI para conocer cual es la correcta implementación de sistemas con RMI
- La arquitectura RPC se utiliza cuando se requiere utilizar métodos almacenados en dispositivos fuera de la red y/o la conexión de manera remota.
- La arquitectura RPC permite el uso compartido de recursos de computación, ya que permite que un dispositivo especializado en una tarea específica como el cálculo de datos numéricos y que no esté en la red de computadoras que tiene el servidor que posee el servicio para calcular los datos sea capaz de utilizarlo.

APLICACIÓN PROFESIONAL DE LA PRÁCTICA REALIZADA.

La implementación de soluciones basadas en RMI y RPC en el mundo profesional se enfoca en permitir el despliegue de arquitecturas cliente servidor las cuales las cuales permitan de forma muy simple y enfocado en sistemas básicos, los cuales busquen satisfacer necesidades como el acceso a bases de datos, obtención de datos de recursos de red y demás elementos requeridos para el funcionamiento de sistemas, permitiendo hasta el control de mecanismos industriales.

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 26 de 28

BIBLIOGRAFIA UTILIZADA.

Amazon Web Services. (2021). *Guía del usuario de Lumberyard* (Amazon Web Services (ed.); 28th ed., Vol. 1).
https://docs.aws.amazon.com/es_es/lumberyard/latest/userguide/lumberyard-ug.pdf#network-replicas-remote-procedure-calls

How, K. (2020, March 24). *Remote procedure call (RPC): ¿en qué consiste esta tecnología?* -. IONOS.
<https://www.ionos.es/digitalguide/servidores/know-how/que-es-rpc/>

Perez, F. (2015). *RPC: Llamada a procedimiento remoto*.
http://laurel.datsi.fi.upm.es/_media/docencia/asignaturas/sd/comunicacion_parte3-1pp.pdf


Cloudflare. (s.f.). ¿Qué es el modelo OSI? Obtenido de <https://www.cloudflare.com/es-es/learning/ddos/glossary/open-systems-interconnection-model-osi/>

Fuentes, F. d. (2014). *Sistemas Distribuidos*. Cuajimalpa: Universidad Autonoma Metropolitana.

Universidad de Alicante. (2004). *Introducción a RMI*. Obtenido de <http://www.jtech.ua.es/j2ee/2003-2004/modulos/rmi/sesion01-apuntes.htm>

RÚBRICA DE EVALUACIÓN DE LA PRÁCTICA

INDIVIDUAL		CRITERIOS DE EVALUACIÓN					NOT A
Habilidad	Estudiante	0 – 1,5	1,6 - 2,9	3,0 - 3,9	4,0 - 4,5	4,6 - 5,0	
	1. Entiende la visión que un sistema Distribuido deberá cumplir sus propiedades y desafíos	No los identifica, no conoce las funciones de cada uno	Identifica algunos, no conoce las funciones	Identifica algunos conoce algunas funciones	Identifica todos, conoce algunas funciones	Identifica todos los mecanismos y conoce sus funciones	

 UNIVERSIDAD DE SAN BUENAVENTURA SEDE BOGOTÁ	FORMATO PARA PRACTICAS DE LABORATORIO			
	PROGRAMA: INGENIERÍA DE SISTEMAS			
	LAB-02	Versión: 1	Fecha: 24/09/2021	Página 27 de 28

	2. Realizar investigaciones utilizando la bibliografía existente.	No utilizó bibliografía	Utilizó bibliografía pero no realizó las citaciones	Utilizó bibliografía a no científica o educativa	La bibliografía reseñada no corresponde con las citas empleadas en la investigación	Uso bibliografía y realizó las citaciones correspondientes de forma adecuada	
	3. Logra expresar ideas propias a partir los conocimientos que adquiere en la investigación.	No utiliza palabras propias ni ideas propias	Las ideas plasmadas son confusas, desordenadas y no corresponden al tema	Las ideas que expresa con coherentes, pero no corresponden al tema	Las ideas son coherentes pero están desordenadas y no logran concluir	Las ideas son coherentes, ordenadas y pertenecen a la temática	
	4. Forma conceptos utilizando las guías conceptuales de forma crítica.	No utiliza palabras propias ni ideas propias	No muestra una interpretación de las ideas investigadas	La interpretación que muestra no corresponde a la temática	La interpretación que muestra solo repite lo leído	La interpretación no se limita a los conceptos investigados, dejando claras sus ideas en torno al tema	
	5. Comunica de forma verbal los resultados obtenidos en su investigación, siendo claros y concretos	No realiza presentación de su investigación	La presentación no contiene todos los conceptos involucrados en la investigación	La presentación está completa pero la expresión verbal no logra transmitir los conocimientos adquiridos	La presentación está completa pero la expresión verbal solo expresa el contenido de la presentación misma (lee la presentación)	La presentación está completa y la expresión verbal logra transmitir los conocimientos adquiridos	
	Total	Total = (N1 + N2 + N3 + N4 + N5) / 5					



**UNIVERSIDAD DE
SAN BUENAVENTURA
SEDE BOGOTÁ**

FORMATO PARA PRACTICAS DE LABORATORIO

PROGRAMA: INGENIERÍA DE SISTEMAS

LAB-02

Versión: 1

Fecha:
24/09/2021

Página **28** de
28