|  |  |
| --- | --- |
|  | |
| **INFORME DE LABORATORIO** | |
| ESTUDIANTES:  * Luis Felipe Velasco Tao * Juan David González Dimaté | **ASIGNATURA: SISTEMAS DISTRIBUIDOS** |
| **GRUPO: 4351** |
| **NOTA:** |
| **CONSULTA PREVIA**   1. Modelo OSI: 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:    1. Aplicación: Interacción directa con los datos del usuario.    2. Presentación: Traducción, cifrado y comprensión de los datos. Prepara los datos para el uso de la capa de aplicación.    3. Sesión: Apertura y cierre de comunicaciones.    4. Transporte: Comunicaciones de extremo a extremo entre dos dispositivos.    5. Red: Posibilitar transferencias de datos entre dos redes diferentes.    6. Enlace de datos: Posibilitar transferencias de datos en una misma red.    7. Física: Dispositivos que participan en la transferencia de datos. 2. 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 arqui- tectura 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. 3. 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, compartiendo recursos y carga de procesamiento a través de varios sistemas. RMI permite exportar objetos como objetos remotos para que otro proceso remoto pueda acceder directamente como un objeto Java. 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 al núcleo que envíe el mensaje al servidor, posteriormente se bloquea hasta que regrese la respuesta. | |
| **FORMULE TRES HABILIDADES QUE DESEE ADQUIRIR O DESARROLLAR A TRAVÉS DE LA PRÁCTICA DE LABORATORIO.** *El estudiante formulará desde la revisión de la guía las habilidades que desarrollará a través*  *de la práctica*   * ***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*** | |
| **Elabore un Mapa conceptual del tema a tratar en la Práctica de Laboratorio.**  *Interfaz de usuario gráfica, Aplicación, Tabla  Descripción generada automáticamente* | |
| **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 obsoltet  //Addition Hello = new Addition();  //Naming.bind("rmi://localhost/ABC", Hello);    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:**    **Servidor en marcha**    ***ClientSide***  **Interface**  package pack;  import java.rmi.\*;  public interface AdditionInterface extends Remote {  public int add(int a, int b) throws RemoteException;  }  **Argumentos**    **Cliente**  package pack;  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 obsolteto  //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**    **Estructura de los proyectos** | |
| **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);  }  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();  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));  jLabel1.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  jLabel1.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()  .addComponent(jLabel2)  .addGap(105, 105, 105)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jLabelGetData, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  .addComponent(jLabelSuma, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_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(jLabel1)  .addGap(320, 320, 320))  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(layout.createSequentialGroup()  .addGap(22, 22, 22)  .addComponent(jLabel1)  .addGap(43, 43, 43)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel2)  .addComponent(jTextFieldData, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_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.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGroup(layout.createSequentialGroup()  .addGap(72, 72, 72)  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)  .addComponent(jLabel3)  .addComponent(jButtonSumar))))  .addGap(2, 2, 2)  .addComponent(jTextFieldNumB, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(18, 18, 18)  .addComponent(jLabelSuma, javax.swing.GroupLayout.DEFAULT\_SIZE, 26, Short.MAX\_VALUE)  .addContainerGap())  );  pack();  }// </editor-fold>  private void jTextFieldNumAActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  }  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 jButtongETdATAActionPerformed(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() {  @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 jButtongETdATA;  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':  pp = P1.indexOf(c) + l + p;  if (pp - L >= 0) {  pp -= L;  }  return P2.charAt(pp);  case 'D':  pp = P2.indexOf(c) - l - 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**     1. Observa la siguiente imagen y realiza la debida implementación del proceso RPC     **Respuesta**   * **De lado del servidor:**   Creamos una interfaz que establece el endpoint y los servicios que se van a utilizar en el programa.    Luego creamos una clase que implementará este endpoint junto a sus servicios disponibles.    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>.   * **De lado del cliente:**   Creamos una clase que va a consumir el servicio web previamente establecido.    Luego de haber realizado lo anterior, ejecutamos el Publisher y luego ingresamos a la URL que le habíamos establecido.  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.     1. 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.    Creamos la implementación del Web Service.    Creamos el Publisher.    Finalmente creamos el Cliente.    Ahora ejecutamos el Publisher, obtenemos los datos para el QName y ejecutamos el programa.     1. 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    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.    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:    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:    Finalmente, se ejecuta el servidor/publicados y el cliente, y se obtiene el siguiente resultado:     1. 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**  Diagrama  Descripción generada automáticamente   1. 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**   1. 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 realizo la llamada. * 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 encontrase 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. | |
| **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** | | | | | **NOTA** | | **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** |  | | **2. Realizar investigaciones utilizando la bibliografía existente.** | **No utilizó bibliografía** | **Utilizó bibliografía pero no realizó las citaciones** | **Utilizó bibliografía 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, desordenas y no corresponden a al tema** | **Las ideas que expresa con coherentes, pero no corresponden al tema** | **Las ideas son coherentes pero están en desorden 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** | | | | |  | | |