|  |  |
| --- | --- |
|  | |
| **INFORME DE LABORATORIO**  (*Para elaborar por el Estudiante)* | |
| ESTUDIANTES:  * Juan David Gonzalez Dimaté * Luis Felipe Velasco Tao | **ASIGNATURA: SISTEMAS DISTRIBUIDOS** |
| **GRUPO: 4351** |
| **NOTA:** |
| **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*   * ***Entender las arquitecturas que se pueden utilizar al momento de plantear o desarrollar un sistema distribuido.*** * ***Reconocer el tipo de arquitectura más adecuado al momento de implementar un sistema distribuido.*** * ***Implementar adecuadamente un sistema distribuido en un sistema con otra arquitectura.*** | |
| **Elabore un Mapa conceptual del tema a tratar en la Práctica de Laboratorio.** *El estudiante construirá un mapa conceptual a partir de las palabras claves suministradas por el docente*  **Interfaz de usuario gráfica  Descripción generada automáticamente con confianza media**  **Las siguientes fuentes fueron las empleadas para realizar el mapa conceptual:**  (Reynoso, 2004)**,** (Santamaría, 2012a)**,**(Departamento de ciencias de la Computación, 2019)**,** (Departamento de Ciencia de la Computación e Inteligencia Artificial, 2005)**,** (Drake, 2017)**,** (Alvarez, 2001)**,** (Llamas, 2003)**,** (De Ves & Cerveron, 2007) | |
| **RESULTADOS.**  De los ejercicios realizados de sockets, seleccionamos el ejercicio del supermercado, en el cual el cliente le enviara  al servidor como petición el total de una compra y recibirá por respuesta el descuento que se le debe aplicar a la  compra. Se obtuvieron los siguientes Scripts desarrollados en Python:  **Servidor**  import tkinter  import socket, threading  win = tkinter.Tk() # Crea la ventana principal  win.title('Servidor del supermercado')  win.geometry("500x500")  users = {}#Diccionario de usuario, también puede conectarse a la base de datos  ip = "192.168.0.14"  tablero = ""  icon = "icons/server.png" # ubicacíón del icono  flag = False  port = 5000  text = tkinter.Text(win, height=25, width=60)  labeltext = tkinter.Label(win, text='LOG DEL SERVIDOR',font=('Helvetica', 18, 'bold')).grid(row=0, column=0)  text.grid(row=1, column=0)  win.iconphoto(False, tkinter.PhotoImage(file=icon))  def run(ck, ca):  global flag  try:  userName = ck.recv(1024)# Aceptar como unidad la información enviada por el cliente en 1k. La información recibida aquí es de tipo byte  users[userName.decode("utf-8")] = ck#Decodificar y almacenar información del usuariofont=('Helvetica', 18, 'bold')  printStr = userName.decode("utf-8") + " conexión\n"#Muestre si la conexión se realizó correctamente en el cuadro de visualización de la conexión  text.insert(tkinter.END, printStr)  while True:  rData = ck.recv(1024)# Aceptar como unidad la información enviada por el cliente en 1k. La información recibida aquí es de tipo byte  dataStr = rData.decode("utf-8")  infolist = dataStr.split(":")#Dividir la cadena para obtener el nombre de usuario que se enviará y la información enviada por el cliente  d = int (infolist[1])  r = ""  ganancia = int(d\*0.15)  total = int(d-ganancia)  r = "- Total sin descuento: $"+str(d)+"\n Descuento (15%): $" + str(ganancia)+"\n TOTAL A PAGAR: $"+str(total)+"\n"  tablero = "- Consulta realizada por "+str(infolist[0])+" - Valor compra $"+str(d)  users[infolist[0]].send((r).encode("utf"))  text.insert(tkinter.END, tablero+"\n")  # El cliente que quiere enviar información envía información al cliente objetivo  flag = False  except:  if flag is not True:  text.insert(tkinter.END, "- El servidor esta apagado\n")# Mostrado en la ventana de información  flag = True  def start():  server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)#socked ipv4 o ipv6 y acuerdos relacionados  server.bind((ip, port))# ¡IP vinculante y número de puerto! ! ! 1: Tenga en cuenta que el número de puerto de entrada es de tipo str y aquí debe pasarse en tipo int  # 2: el parámetro de bind () tiene la forma de una tupla  server.listen(10)# Configure el monitoreo y establezca el número máximo de conexiones  text.insert(tkinter.END, "- El servidor se inició correctamente\n")# Mostrado en la ventana de información  while True:# El bucle infinito se usa aquí porque el servidor simulado debe estar ejecutándose todo el tiempo  ck, ca = server.accept()# Aceptar la información del cliente conectado  # Donde ca es una tupla compuesta por ip y número de puerto, ck información sobre el cliente  t = threading.Thread(target=run, args=(ck, ca))#Abre un hilo cada vez que un cliente está conectado  # Los parámetros de la función entrante en la función Thread también están en forma de tuplas  t.start()#Hilo abierto  def startSever():  s = threading.Thread(target=start)#Habilitar un hilo para iniciar el servidor  s.start()#Hilo abierto  # Lo siguiente trata sobre el funcionamiento de la interfaz  startSever()  win.mainloop()  La parte en verde define la actividad del servidor al realizar el cálculo del ahorro del cliente al realizar una compra con cierto valor.  **Cliente**  import tkinter  import socket  import threading  from random import randint  win = tkinter.Tk()  idt = str(randint(0,50))  win.title("Cliente Supermercado N° "+idt)  win.geometry("500x500")  ip = "192.168.0.14"  port = 5000  cliente\_id = "Cliente "+idt  icon = "icons/market.png" # ubicacíón del icono  estadoVar = ""  flag = False  ck = None# Se utiliza para almacenar información del cliente  def getInfo():  global flag  global ck  while True:  try:  data = ck.recv(1024)#Se utiliza para recibir la información enviada por el servicio  text.insert(tkinter.END, data.decode("utf-8"))#Display en el cuadro de mensaje  flag = False  except:  ck = None  if flag is not True:  estadoVar = "Sin conexion"  text.insert(tkinter.END,"- "+estadoVar+"\n")  flag = True  def connectServer():  global ck  try:  client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)#socked ipv4 o ipv6 y acuerdos relacionados  client.connect((ip, port))# ¡Conecta ip y número de puerto! ! ! 1: Tenga en cuenta que el número de puerto de entrada es de tipo str y aquí debe pasarse en tipo int  # 2: el parámetro de bind () tiene la forma de una tupla  client.send(cliente\_id.encode("utf-8"))  ck = client  t = threading.Thread(target=getInfo)  estadoVar = "Conectado"  text.insert(tkinter.END,"- "+estadoVar+"\n")  t.start()  except:  estadoVar ="Sin conexion"  text.insert(tkinter.END,"- "+estadoVar+"\n")  def reconectar():  global flag  if flag is True:  connectServer()  def pedirPrestamo():  st = salario.get()  sendStr = cliente\_id+":"+st  ck.send(sendStr.encode("utf-8"))  salario.set("")  #Abajo está la interfaz  win.iconphoto(False, tkinter.PhotoImage(file=icon))  salario = tkinter.Variable()  labeltext= tkinter.Label(win, text="DESCUENTO 15%", font=('Helvetica', 18, 'bold')).grid(padx=5,pady=5,row=0, column=0, columnspan=2)  labelUse = tkinter.Label(win, text="ID: ").grid(padx=5,pady=5,row=1, column=0)  entryUser = tkinter.Label(win, text=cliente\_id).grid(padx=5,pady=5,row=1, column=1,sticky="ew")  Salario = tkinter.Label(win, text="Valor compra: $").grid(padx=5,pady=5,row=2, column=0, sticky="ew")  labeltext= tkinter.Label(win, text="Respuesta",font=('Helvetica', 18, 'bold')).grid(padx=5,pady=5,row=4, column=0, columnspan=2)  button1 = tkinter.Button(win, text="Enviar", command=pedirPrestamo, bg="#62DF9B").grid(padx=5,pady=5,row=3, column=0,columnspan=2, sticky="ew")  entrySalario = tkinter.Entry(win, textvariable=salario).grid(padx=5,pady=5,row=2, column=1,sticky="ew")  text = tkinter.Text(win,pady=5, height=10, width=60)  text.grid(padx=5,pady=5,row=5, column=0,columnspan=2)  connectServer()  button2 = tkinter.Button(win, text="Reconectar", command=reconectar, bg="#DF6264").grid(padx=5,pady=5,row=7, column=0,columnspan=2, sticky="ew")  win.mainloop()  El cliente será el que se ubicará en Linux en unja máquina virtual y accederá de forma remota al servidor ubicado en Windows.  Cliente ubicado en máquina virtual con Sistema operativo Kali Linux    Servidor ubicado en maquina Host con sistema operativo Windows 10:    A continuación se puede ver en funcionamiento el socket con su servidor en Windows 10 y con un cliente en Kali Linux, haciendo uso del puerto 5000:    Se comprueba el uso de los puertos en Windows mediante el comando nestat:    Aquí se logra evidencia que, la maquina virtual con Hostname “Tao” se esta comunicando  con la maquina host Windows designada por la IP bajo el protocolo TCP. Por otro lado, se  reviso el uso de puertos en la máquina virtual:    En el caso del servidor, se debe buscar el puerto al cual este tiene acceso del servidor, lo cual  se realizó con el comando **ss** que hace un barrido por las conexiones existentes de la maquina  y se filtra el contenido por medio de la instrucción grep y la cadena con la IP del servidor. | |
| **CUESTIONARIO.**   1. Explicar 5 aplicaciones que funcionan en el modelo cliente/servidor:    1. Servidor de correo: Un usuario puede utilizar un cliente para realizar diversas operaciones para otros clientes dentro del servidor en la red.    2. Servidor de archivos: Un cliente puede almacenar un archivo dentro del servidor para poder distribuirlo con los demás clientes.    3. Servidor de bases de datos: Desde un cliente se pueden almacenar datos en la base de datos, mientras que en otro cliente se pueden acceder a esos datos, siempre y cuando correspondan al mismo servidor.    4. Servidor web: Las páginas web tienen como cliente al navegador mientras que este utiliza un servidor Apache.    5. Servidor de impresión: Los usuarios de una red de impresoras pueden realizar una solicitud para imprimir un documento en una determinada impresora al servidor, que le transfiere dicha solicitud, junto al documento a la impresora. 2. Cuál es el concepto, función de socket: Es un túnel de comunicación que ayuda a que 2 aplicaciones se comuniquen. Son los puntos finales de enlaces de comunicaciones entre procesos (Oracle, 2021). 3. Cómo funciona un socket:  Los procesos tratan a los sockets como descriptores de ficheros, de forma que se puedan intercambiar datos con otros procesos, transmitiendo y recibiendo a través de ellos. Dependiendo de su tipo se describe una forma distinta para transferir información (Oracle, 2021):    1. Stream Socket: Son un servicio orientado a conexión donde los datos se transfieren sin encuadrarlos en registros o bloques. El protocolo de comunicaciones con streams es un protocolo orientado a conexión, ya que para establecer una comunicación utilizando el protocolo TCP, hay que establecer en primer lugar una conexión entre un par de sockets. Mientras uno de los sockets atiende peticiones de conexión (servidor), el otro solicita una conexión (cliente).    2. Datagram Socket: Son un servicio de transporte sin conexión. Es más eficiente que TCP, pero no garantiza la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no está garantizada. Los paquetes pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió. Debido a que es un protocolo sin conexión, cada vez que se envían datagramas es necesario enviar el descriptor del socket local y la dirección del socket que debe recibir el datagrama. 4. Que es un modelo peer to peer, cita un ejemplo aplicativo:   Se refiere a una red que no tiene clientes y servidores fijos, sino una serie de nodos que se comportan a la vez como clientes y servidores de los demás nodos de la red (Salitas, 2008). Un ejemplo aplicativo de este modelo es la telefonía IP, como Skype, que permite aprovechar los recursos de la red para establecer una conexión entre dos nodos, siendo uno de ellos el cliente y el servidor del otro al mismo tiempo y viceversa.   1. Explica la estructura por medio de un ejemplo aplicativo del modelo hibrido o centralizado:   La primera generación de P2P (Napster) empleaba una estructura de red cliente-servidor. El servidor central mantiene una base de datos con información de los ficheros servidos por cada par. Cada vez que un cliente se conecta o desconecta de la red, la base de datos se actualiza. Todos los mensajes de búsqueda y control son enviados al servidor centralizado. El servidor centralizado compara la solicitud de sus clientes con el contenido de su base de datos y envía las correspondencias al cliente en cuestión. Una vez que es informado de las correspondencias, el cliente contacta con el par directamente y accede al recurso solicitado.   1. Explica la estructura por medio de un ejemplo aplicativo del modelo puro o totalmente descentralizado.   La segunda generación de P2P (Gnutella) usa un modelo distribuido donde no existe ningún servidor central y todos los nodos tienen el mismo estatus. Cada nodo actúa como servidor y como clientes en la red. Como es evidente cada par dentro de esta arquitectura trata de mantener un cierto número de conexiones con otros pares durante todo el tiempo. Este conjunto de pares conectados transporta el tráfico de red, que está conformado esencialmente por peticiones y respuestas a esas peticiones, y varios mensajes de control que facilitan el descubrimiento de otros nodos (mensajes ping en Gnutella). A pesar de que el número de saltos de la red es potencialmente infinito, permanece limitado por un tiempo de vida máximo o TTL, relacionado con el máximo número de saltos que puede dar un mensaje. Por cada nodo o par por el que circula el mensaje de petición se decrementa en una unidad el TTL descartándose la petición si esta llega a cero. El modelo P2P puro es más robusto al no depender de un servidor central, además es más económico. La principal desventaja es el elevado tiempo y sobrecarga de ancho de banda que suponen las búsquedas de información en la red. Además, puede ser que el recurso buscado y existente ni siquiera pueda ser encontrado.   1. Explica la estructura por medio de un ejemplo aplicativo del modelo mixto o semi descentralizado:   Dentro de este modelo, ciertos pares de la red son seleccionados como superpares y ayudan a gestionar el tráfico dirigido hacia otros pares. Los superpares cambian dinámicamente a medida que nuevos pares se conectan.  En este modelo cada nodo cliente mantiene sólo un pequeño número de conexiones abiertas y cada una de esas conexiones es a un superpar. Así mismo los superpares están conectados entre sí. Esta nueva topología virtual tiene el efecto de hacer la red escalable, mediante la reducción del número de nodos involucrados en el encaminamiento y manejo de los mensajes, así como la disminución del volumen de tráfico entre ellos. Además, la velocidad de respuesta a las solicitudes dentro de un entorno mixto es comparable al de un entorno P2P centralizado.   1. Realice un Comparativo entre un sistema cliente Servidor de Dos capas Vs. Uno de tres capas.  |  |  |  | | --- | --- | --- | | Arquitectura. | Dos niveles | Tres niveles | | Definición | Sistemas cliente/servidor en donde el cliente solicita recursos y el servidor responde directamente a la solicitud con sus propios recursos. | Sistemas cliente/servidor en donde existe un intermediario. El cliente solicita recursos, el intermediario proporciona los recursos solicitados mediante otro servidor y el servidor de datos proporciona al intermediario los datos que requiere.  Cada servidor realiza una tarea especializada (un servicio). Por lo tanto, un servidor puede utilizar los servicios de otros servidores para proporcionar su propio servicio. | | Ventajas | Es polivalente.  Puede responder directamente a todas las solicitudes de recursos del cliente. | Mayor grado de flexibilidad.  Mayor seguridad.  Mejor Rendimiento. |  1. ¿Qué características debe tener el motor de base de datos al montar el sistema de información?   Se debe tener en cuenta que una base de datos en un sistema distribuido debe estar formada por varias computadoras en donde se deben encontrar fragmentos o réplicas de la base de datos principal, además de poseer el sistema gestor de la base de datos y el software requerido para la comunicación con los otros nodos. Sin mencionar que debe existir la red de comunicaciones para este propósito. Cada uno de los ordenadores que forman parte de la red deben tener autonomía local. Finalmente, ningún nodo debe tener un rango superior a los demás, para evitar una centralización del sistema (Date, 2001).   1. ¿Si se tiene una base de datos es distribuida cuantos servidores de aplicaciones se deben tener? Argumente la respuesta. 2. Crea un cuadro comparativo entre los diferentes modelos de cliente servidor cuales son las ventajas y desventajas de los diferentes modelos (LOPEZ FUENTES, 2015)  |  |  |  |  | | --- | --- | --- | --- | | Arquitectura. | Dos niveles | Tres niveles | N Niveles | | Definición | Sistemas cliente/servidor en donde el cliente solicita recursos y el servidor responde directamente a la solicitud con sus propios recursos. | Sistemas cliente/servidor en donde existe un intermediario. El cliente solicita recursos, el intermediario proporciona los recursos solicitados mediante otro servidor y el servidor de datos proporciona al intermediario los datos que requiere. | Cada servidor realiza una tarea específica, por lo que un servidor del sistema puede usar los servicios de otro servidor para poder proporcionar su propio servicio. | | Ventajas | Es polivalente.  Puede responder directamente a todas las solicitudes de recursos del cliente. | Mayor grado de flexibilidad.  Mayor seguridad.  Mejor Rendimiento. | Facilita la integración entre diferentes sistemas.  Mayor interacción con el usuario. | | Desventajas | Poca seguridad. | Costes elevados. | Dificultad de reparación.  Costes elevados. | | |
| **CAUSAS DE ERROR Y ACCIONES PARA OBTENER MEJORES RESULTADOS**   * **La instalación de Python en Kali Linux se nos dificulto, al igual que el uso de la librería Tkinter.** * **Manejo de interfaz gráfica con Tkinter**   **Estos Items generaron cierto grado de dificultad, pero se solucionaron a tiempo. Los siguientes Items son acciones y causas de error con relación a los conceptos manejados en el temario.**   * **Las transacciones dentro del sistema distribuido no sean atómicas.** * **Algún nodo del sistema tenga un nivel jerárquico, que pueda provocar algún tipo de centralización.** | |
| **CONCLUSIONES.** *El estudiante consignará en este espacio la síntesis de los logros alcanzados a través del desarrollo de la práctica de Laboratorio.*   * **El uso de sockets es de gran utilidad para comprender el funcionamiento básico de un sistema distribuido** * **La arquitectura Cliente-Servidor requiere de habilidades en el diseño y especificación de actividades de los componentes** | |
| **APLICACIÓN PROFESIONAL DE LA PRÁCTICA REALIZADA.** *Exponga en este espacio la relación de las competencias y habilidades desarrolladas con el ejercicio de su futura profesión.*  En el ámbito de las bases de datos, los sistemas distribuidos que permiten el acceso a la información desde determina- dos servidores está cada vez más extendido. Un ejemplo de esto puede ser los servicios de búsqueda que ofrece Google, ya que por medio de su navegador “Google Académico”, permite acceder a las distintas bases de datos aloja- das en distintos servidores alrededor del mundo, desde un solo cliente en uso.  El uso de Sockets y arquitecturas cliente-servidor es de vital importancia en el mundo laboral debido a que, sobre estos conceptos, se construyen la mayoría (por no decir que todos) las páginas y aplicaciones web, móvil y servicios web a los cuales podemos tener acceso con tener conexión a internet, y en la actualidad, con todos los negocios que han migrado a la digitalizad se requiere de profesionales capacitados en el diseño de sistemas aptos para que se desarrollen las actividades comerciales con facilidad. | |
| **BIBLIOGRAFIA UTILIZADA.** *Utilizando las normas APA*  Alvarez, F. (2001). *Objetos Distribuidos y Agentes Móviles (Parte I)* [UNIVERSIDAD DE OVIEDO]. http://di002.edv.uniovi.es/~falvarez/cursodoctorado.pdf  De Ves, E., & Cerveron, V. (2007). *Conceptos sobre procesamiento de transacciones Tema 3: Bases de Datos II Contenidos del tema 3*. http://informatica.uv.es/iiguia/2000/BD2/3\_0\_BD2Tema3\_06.pdf  Departamento de Ciencia de la Computación e Inteligencia Artificial. (2005, May 17). *Tema 7: Objetos Distribuidos*. Universidad de Alicante. http://www.jtech.ua.es/historico/paj/restringido/apuntes/sesion13-apuntes.htm  Departamento de ciencias de la Computación. (2019). *Sistemas Distribuidos Módulo 1*. http://cs.uns.edu.ar/~sd/data/apuntes/SD-2019-mod 02.pdf  Drake, J. . (2017). PROGRAMACION CONCURRENTE II.2 Invocación remota de procedimientos. In *II.2 Invocación remota de procedimientos*. https://www.ctr.unican.es/asignaturas/procodis\_3\_ii/doc/procodis\_2\_02.pdf  Llamas, C. (2003). *Modelos de sistema - 2*. https://www2.infor.uva.es/~cllamas/sd/temasPDF/Capitulo2.pdf  LOPEZ FUENTES, F. D. A. (2015). *Sistemas distribuidos*. Universidad Autónoma Metropolitana, Cuajimalpa. www.cua.uam.mx  Oracle. (2021, January 15). *What Is a Socket?* . Java Documentation. https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html  Reynoso, C. B. (2004). *Introducción a la Arquitectura de Software*. http://carlosreynoso.com.ar/archivos/arquitectura/Arquitectura-software.pdf  Salitas, R. (2008). *Peer 2 Peer Sistemas Operativos Distribuidos*. https://www.academia.edu/5263286/Peer\_2\_Peer\_Sistemas\_Operativos\_Distribuidos  Santamaría, R. (2012a). *Sistemas Distribuidos Modelos*. https://docentes.uaa.mx/guido/wp-content/uploads/sites/2/2015/10/Sistemas-Distribuidos-modelos.pdf  Santamaría, R. (2012b). *Sistemas Distribuidos Replicación*. http://vis.usal.es/rodrigo/documentos/sisdis/teoria/8-replicacion.pdf  C. . Date, Introducción a los sistemas de bases de datos, 7th ed., vol. 1. 2001  Rúbrica para la 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 los servicios de la Interconexión de Redes (internetworking) de datos que involucra diversas redes con diferentes tecnologías de forma transparente para el usuario** | **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** | | | | |  | | |