



**Instituto Politécnico
Nacional
Unidad Profesional
Interdisciplinaria en
Ingeniería y Tecnologías
Avanzadas**

Sistemas Distribuidos

Grupo: 2TV7

Nombres de los alumnos:

Barragán Cruz Nad-Xelly Montserrat

Mendoza Sánchez Javier Alberto

Docente:

Mata Rivera Miguel Félix

Fecha: 07 de octubre del 2025

UPIITA-IPN-TELEMÁTICA

Practica 2 de Sistemas Distribuidos

Mecanismos de comunicación: Sockets

NOMBRE ALUMNO: Barragán Cruz Nad-Xelly Montserrat /Mendoza Sánchez Javier Alberto

GRUPO: 2TV7

1) El paquete interfaz EchoInt (o paquete rmi conceptualmente en java)

Consta del archivo: **EchoInt.java**, el cual **describe** el servicio "echo". Su finalidad es proporcionar a este servicio *una interfaz de invocación a objeto remoto*, que se pueda llamar desde algún punto en la red, **ocultando el hecho de que la comunicación se realiza mediante sockets, de hecho da la "impresión" de que se realiza de forma local (concepto de TRANSPARENCIA en Sistemas Distribuidos)**. Cualquier excepción de comunicación con los sockets debe ser reconvertido a esta excepción.

2) El paquete servidor (server)

Contiene dos archivos EchoServer.java y EchoObject.java:

para el caso del archivo **EchoObject.java**: *implementa* la interfaz *EchoInt* del paso 1 y proporciona el servicio de "echo" (es decir, las instrucciones que son el cuerpo del código).

La implementación de este **servicio** consiste en devolver la cadena de texto que se envía, agregando la URL y la hora de la máquina servidor al cabo de 3 segundos. Este retraso es para simular que el servicio tiene un tiempo de procesamiento largo y apreciable (como ocurre en tareas de Sistemas Distribuidos).

EchoServer.java: es la plantilla (esqueleto) de un servidor secuencial que realiza las siguientes operaciones:

- *Recibe una conexión a través de un socket*
- *Invoca (llama) un objeto de la clase EchoObject.java*
- *Devuelve la respuesta de la anterior invocación por el socket*

NOTA: Existe también una segunda versión multihilo de *EchoServer.java* denominada **EchoMultiServer.java** que se analizará en la segunda parte de esta practica.

El paquete cliente (client)

Consta, de dos archivos:

- **Echo.java**: es el cliente propiamente dicho. Realiza el siguiente ciclo:
 - Leer de teclado
 - Invocar el *stub*
 - Imprimir el resultado por pantalla.
- **EchoObjectStub.java**: Es el *stub* del cliente; es el **proxy** del objeto en el nodo del cliente. Nótese que implementa la misma interfaz que el objeto: interfaz *EchoInt* y, adicionalmente, el método *setHostAndPort*, para especificar con que *host* y *port* se van a realizar las conexiones.

Ejecutamos los archivos:

```
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> javac .\rmi\EchoInt.java
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> javac .\server\EchoObject.java
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> javac .\client\Echo.java
```

Ejecutamos el comando java client.Echo

```
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> java client.Echo
```

Observamos resultados

```
Cliente Echo - Versión NO distribuida
Escribe mensajes:
hola
Procesando: 'hola'
Procesamiento de 'hola'terminado.
NadXelly:09:53> hola (retrasada 3 segundos)
Procesando: 'null'
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> java client.E
cho
Cliente Echo - Versión NO distribuida
Escribe mensajes:
que tal
Procesando: 'que tal'
Procesamiento de 'que tal'terminado.
NadXelly:09:54> que tal (retrasada 3 segundos)
prueba
Procesando: 'prueba'
Procesamiento de 'prueba'terminado.
NadXelly:09:54> prueba (retrasada 3 segundos)
```

II.- EJERCICIOS PRACTICOS

El código base de esta practica esta en el enlace Github:

Primero se desarrollará una versión no distribuida de la práctica con “llamada local a procedimiento”. (*Recuerde que los términos siguientes son sinonimos: procedimiento= subrutina= función= servicio*)

Se asumirá que el cliente y el servidor se encuentran en la misma máquina y el programa cliente invoca los servicios mediante la llamada usual que se hace para un método Java.

En esta versión no distribuida, no existen stubs.

Realice lo siguiente:

- 1) Genera una estructura de carpetas (paquetes como sigue) también puedes generar un proyecto en su entorno IDE (e..g NetBeans o Eclipse) de tu preferencia.
 - a. *Paquete rmi:* archivo *EchoInt.java* .
 - b. *Paquete server:* archivo *EchoObject.java*.
 - c. *Paquete client:* archivo *Echo.java*.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> java server.EchoServer
NadXelly: EchoServer esta escuchando en el puerto: 1007
Procesando: 'hola'
Procesamiento de 'hola'terminado.
Procesando: 'que tal '
Procesamiento de 'que tal 'terminado.
Procesando: 'prueba'
Procesamiento de 'prueba'terminado.
|
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2> java client.Echo localhost 1007
hola
NadXelly:15:05> hola (retrasada 3 segundos)
que tal
NadXelly:15:05> que tal (retrasada 3 segundos)
prueba
NadXelly:15:20> prueba (retrasada 3 segundos)
|
```

- 2) Incluya (Codifique) en este archivo una invocación local. 5) Escriba el código necesario y ejecute la aplicación.

(Versión distribuida)

Para el reporte incluya (codifique) los stubs necesarios en cada paquete y desarrolle la versión distribuida de la aplicación (ver Figura 2 como referencia):

- 6) Genera el paquete *server*:
- 7) Copie el archivo *EchoServer.java*
- 8) Genera el paquete *client*
- 9) Copie el archivo *EchoObjectStub.java*
- 10) **Realice estas modificaciones que están en el código en modo comentario:**

a. Generar una instancia del stub

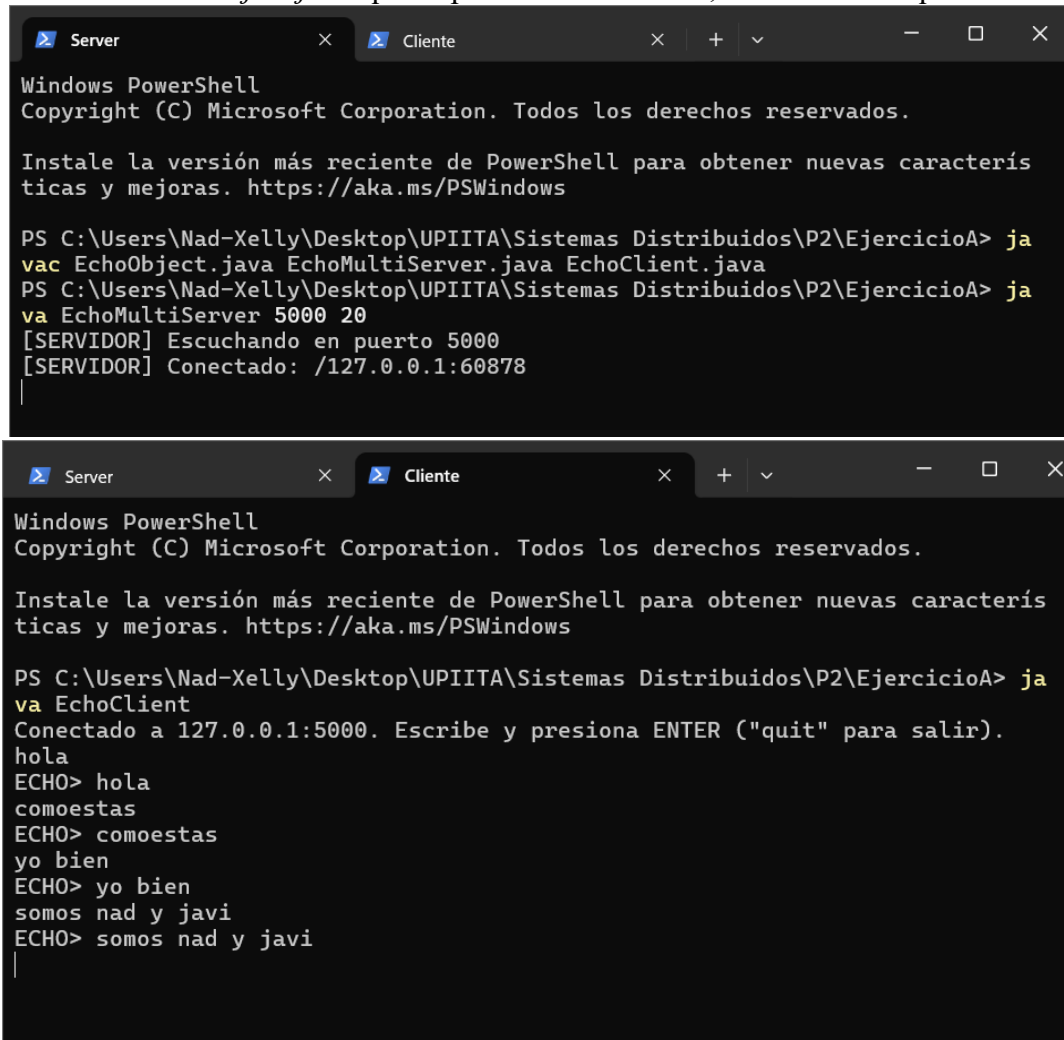
- b. declarar un bucle infinito
- c. Leer del teclado
- d. Invocar (llamar) al stub
- e. Imprimir en pantalla

Ejercicio A : Servidor de echo multihilo.

Un servidor de echo multihilo es un servidor que debe ser *capaz de atender varias peticiones concurrentemente*. La ejecución concurrente de las diferentes peticiones se puede observar creando varios clientes e *iniciando peticiones de servicio desde todos ellos de forma simultánea*. La duración de tres segundos para la ejecución del servicio permitirá observar que las ejecuciones se sobreponen en el tiempo.

Para la realización de este *servicio*, **debe realizarse un nuevo esqueleto del servidor** cuya implementación parcial se proporciona en el archivo **EchoMultiServer.java**. que sustituirá al antiguo esqueleto monohilo del servidor *EchoServer.java*.

El archivo *EchoObject.java* que implementa el servicio, será el mismo que en el caso anterior.



```
Server x Cliente x + v - □ x
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2\EjercicioA> java
vac EchoObject.java EchoMultiServer.java EchoClient.java
PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2\EjercicioA> java
va EchoMultiServer 5000 20
[SERVIDOR] Escuchando en puerto 5000
[SERVIDOR] Conectado: /127.0.0.1:60878
|

Server x Cliente x + v - □ x
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Nad-Xelly\Desktop\UPIITA\Sistemas Distribuidos\P2\EjercicioA> java
va EchoClient
Conectado a 127.0.0.1:5000. Escribe y presiona ENTER ("quit" para salir).
hola
ECHO> hola
comoestas
ECHO> comoestas
yo bien
ECHO> yo bien
somos nad y javi
ECHO> somos nad y javi
|
```