andres ramirez - Pedro Ortiz

SEMINARIO DE ACTUALIZACION

envío de objetos con RMI

Tabla de contenido

[RMI (Java Remote Method Invocation) 2](#_Toc477967208)

[BITACORA DE RED: 2](#_Toc477967209)

[API: 3](#_Toc477967210)

[Acciones\_rmi.jar 5](#_Toc477967211)

[Interfaces 5](#_Toc477967212)

[clienteRMIRed: 7](#_Toc477967213)

[Interfaz gráfica: 11](#_Toc477967214)

[RMIServer: 12](#_Toc477967215)

[Interfaz grafica 15](#_Toc477967216)

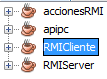
# **RMI (Java Remote Method Invocation)**

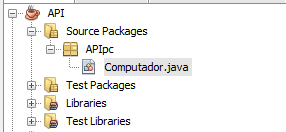
RMI (Java Remote Method Invocation) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI.

RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia, recolección de basura distribuida y paso de tipos arbitrarios.**[[1]](#footnote-1)**

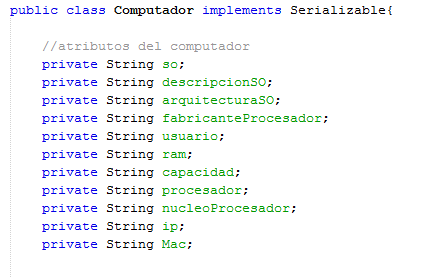
## BITACORA DE RED:

La idea del proyecto es poder compartir la información técnica de múltiples computadores(clientes) a un servidor. Esto para ayudar a controlar los cambios que puedan efectuarse en cada uno de ellos. Como es de suponerse, se realizaron tres proyectos por separado:



******API:** Va a contener la clase “Computador” con sus respectivos atributos. Se realiza de manera independiente para utilizarlo como un API tanto en el servidor como en el cliente. Es necesario tenerlo en todos los computadores que quieran compartir sus especificaciones.

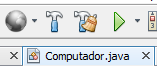
Después de crear la clase “Computador.java” creamos atributos que vamos a compartir por medio de RMI. Además, se debe implementar la interfaz serializable que va a permitir enviar por medio de la red, objetos de este tipo. Nota: insertar get y set.



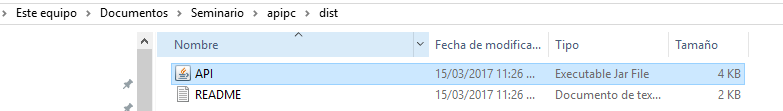
Para que los computadores que van a usar el aplicativo reconozcan el tipo de objeto que se va a enviar por la red, es necesario agregar este API creado anteriormente a los proyectos que se va a desarrollar.

Primero se genera el .jar para poder insertarlo en clienteRMIRed y servidor\_rmi. Así:

* Para generar el JAR simplemente se hace clic sobre el botón:



* El archivo generado va a estar en la ruta “…\nombre\_proyecto \dist”



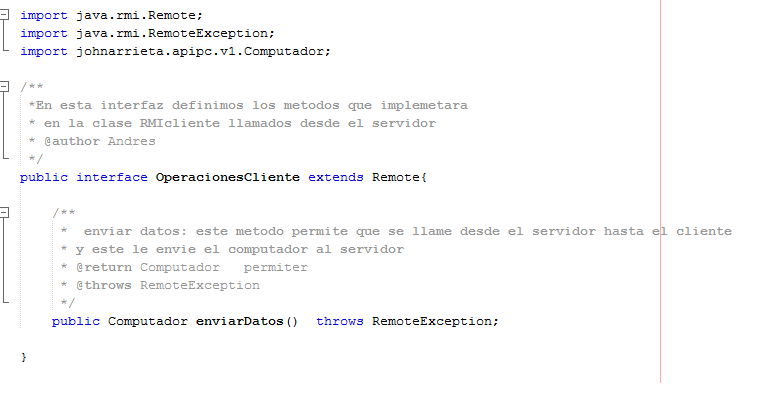
# Acciones\_rmi.jar

Tenemos el contenido del jar exportado a nuestros proyectos.

## Interfaces

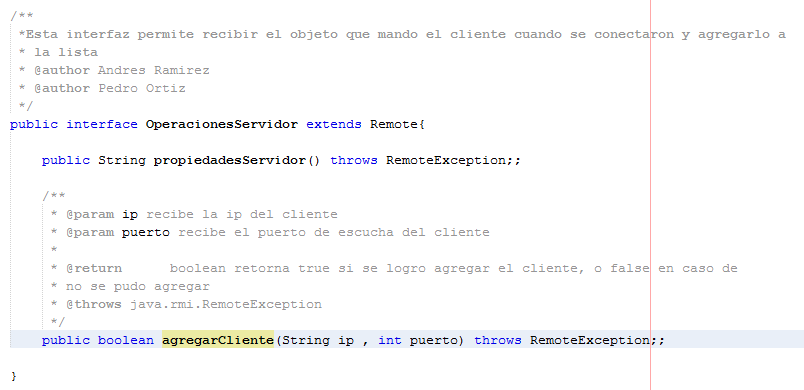
Para la creación del proyecto fue necesario la cración de 2 interfaces donde se estipularon los métodos ejecutar los objetos que implementen dichas interfaces de manera remota.

* Acontinuación la interface que permite llamar métodos remotos del *cliente****.***



En esta clase observamos e método enviarDatos y su respectiva documentación, para ayudar a la compresión de la interface.

A continuación crearemos la interface que nos permita realizar las operaciones sobre el servidor el cual tendrá un método agregar cliente y tenemos la respectiva documentación esta y que se necesita para implementarla en el servidor



Clase Ahora tenemos la Clase Utilidad que nos permite indicarle al compilador que se va a realizar una compilación a través de RMI, además de indicarle cual es el tipo de clase a compilar

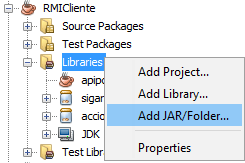


# RMICliente:

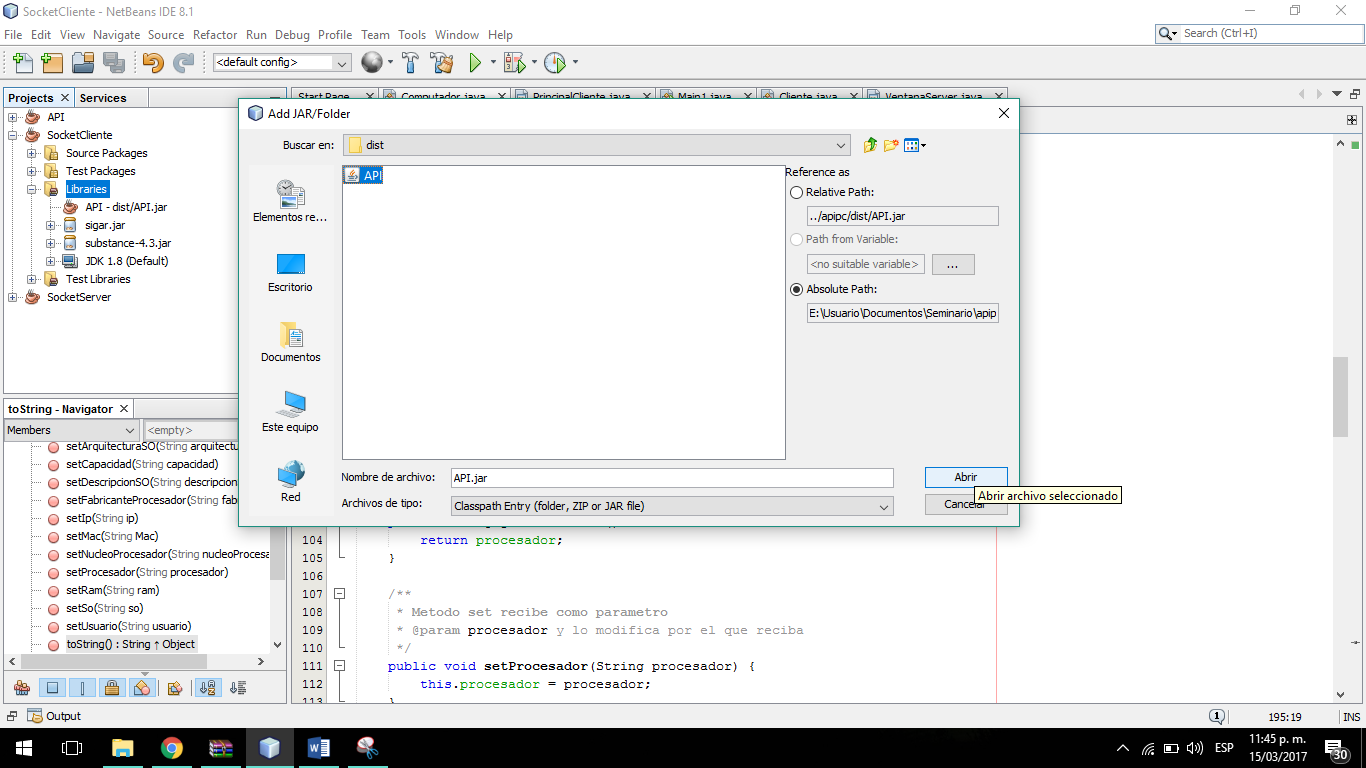
Implementación por medio de RMI para compartir la información hacia el servidor.

Lo primero que debemos realizar es agregar el API de computadores al proyecto de cliente, de la siguiente forma:

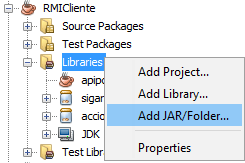
* Add JAR



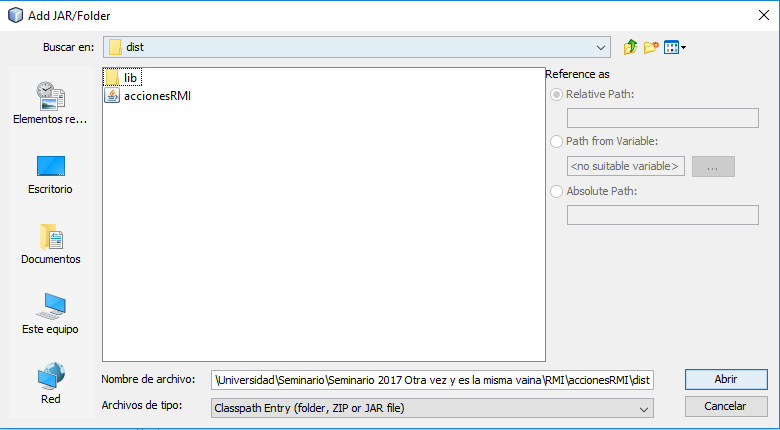
* Buscamos la carpeta dist del proyecto del API y agregamos el JAR generado anteriormente



* Acciones\_rmi.jar debemos agregar este .jar que es el que permite que se llamen métodos remotos aquí en el cliente, y se agrega de la siguiente manera:
* Add JAR

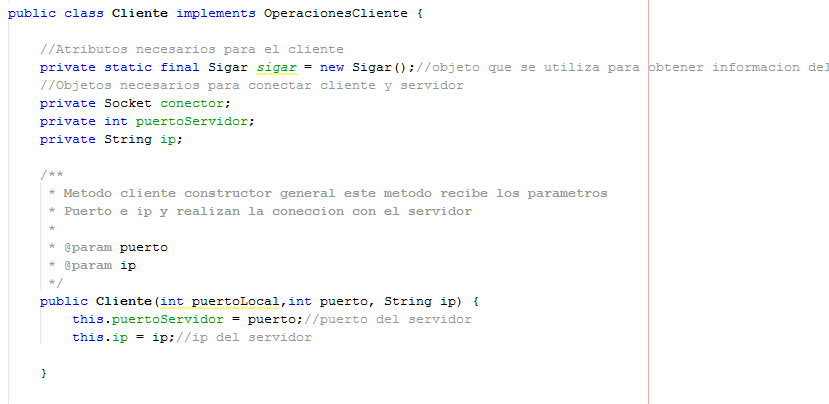


* Luego se busca dentro de la carpeta dist de accionesRMI donde encontraremos el siguiente .jar que debemos seleccionar

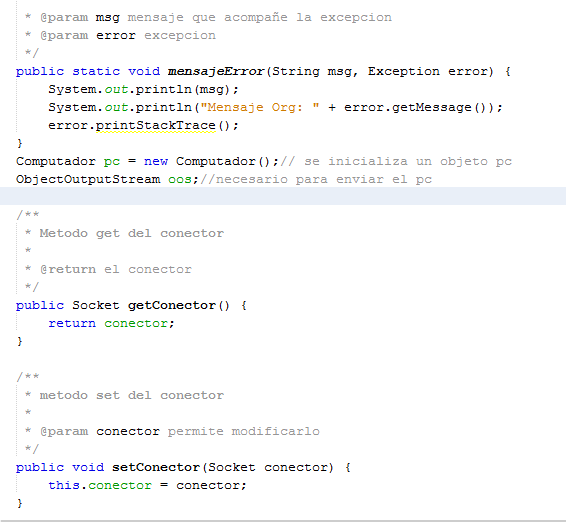


Nota: Además del API se agregaron dos .JAR, uno que mejora la interfaz gráfica del proyector (subtance 4.3) y otro que va a ayudar a extraer información física y/o lógica del computador (sigar).

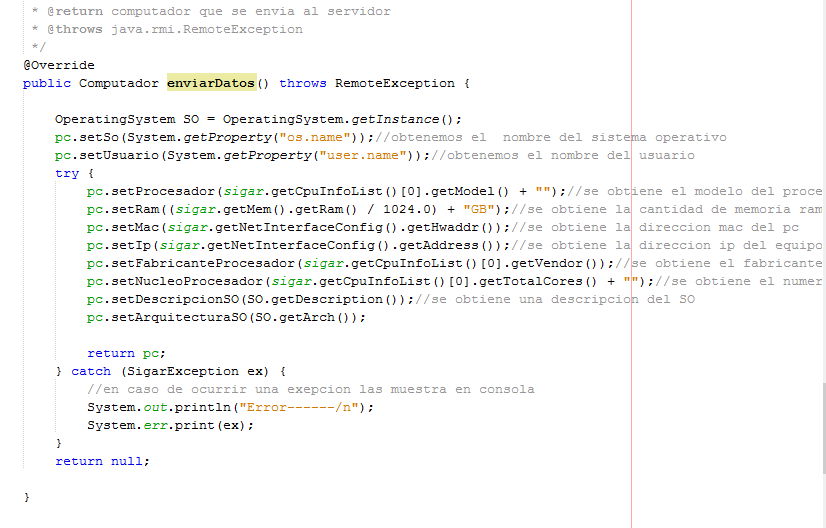
A continuación, se muestra una captura de la implementación de la Clase “Cliente” y que realiza cada sentencia:



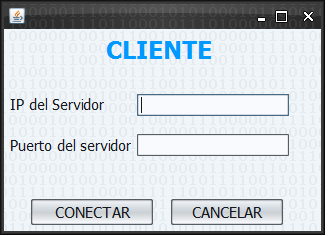
Aquí tenemos los atributos pc que es un computador, además el atributo oos que es el que utilizaremos para enviar el objeto



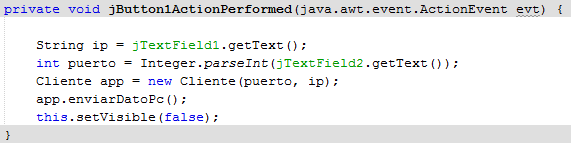
Continuando con la clase Cliente, debemos implementar el método de la interfaz enviar datos, además se obtiene la información del computador y proceda a enviarlo al servidor. Con la librería SIGAR obtenemos lo datos necesarios del computador. Es importante recalcar que los métodos get y set deben ser incluidos en esta clase. La explicación está detallada en la imagen.



Interfaz gráfica: Esta interfaz es muy sencilla, con dos campos de textos donde se introducen la IP y el puerto del servidor. Este debe ser proporcionada por el mismo servidor. Además, dos botones, uno anula la operación y otro que inicia la comunicación.



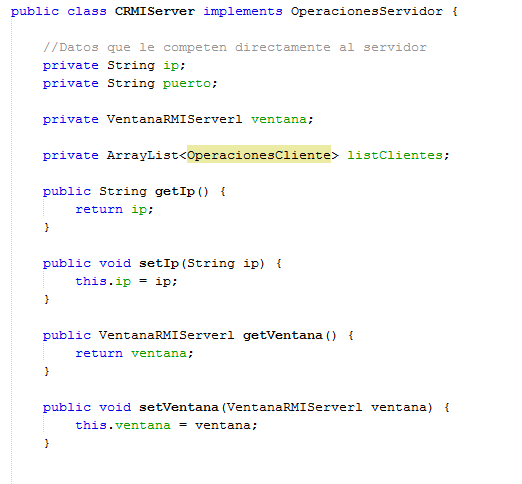
Al accionar el botón conectar se implementa las siguientes líneas de código. S obtienen la ip y puerto de los campos de textos. Se crea el objeto de tipo cliente con sus datos y se envía por el método “enviar dato” implementado anteriormente.



# RMIServer:

Realiza la comunicación y obtiene información de computadores conectados.

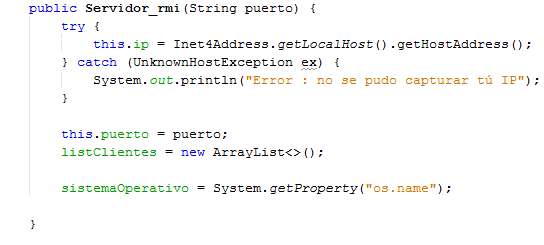
*Clase* ***Servidor\_rmi***



Lo primero que notamos es que implementa la interfaz OperacionesServidor que se encuentra en acciones\_rmi.jar que anteriormente se mostró como incluirla en el proyecto, también tenemos la ip y el puerto del servidor, adicional una lista de Operaciones, también tenemos el atributo VentanaPrincipal que es el frame que permite gestionar la acciones del servidor, hay un lista de clientes ArrayList<OperacionesClientes> lista cliente, la cual hace referencia a los objetos que se exportan por el cliente y permite realizar invocaciones remotas



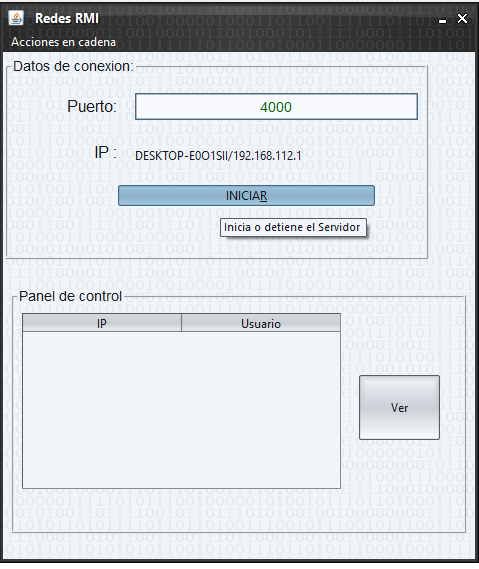
Podemos notar que implementamos el método agregar cliente que teníamos en la interfaz la cual divide la información en dos partes, la ip del cliente y el puerto de conexión y si se agrega un nuevo cliente se actualiza directamente en la tabla, en caso de que ocurra alguna excepción es capturada en el try catch.



Este método lo que hace es capturar la ip del host donde se está ejecutando el servidor, y se activa recibiendo el puerto que el usuario le halla mandado.

### Interfaz grafica

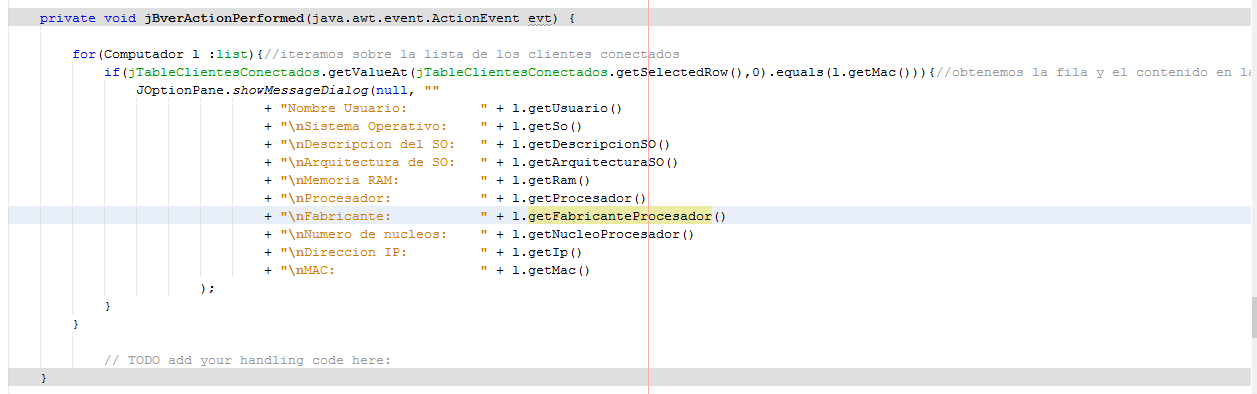
Al iniciar el servidor, notamos que nos carga un panel así



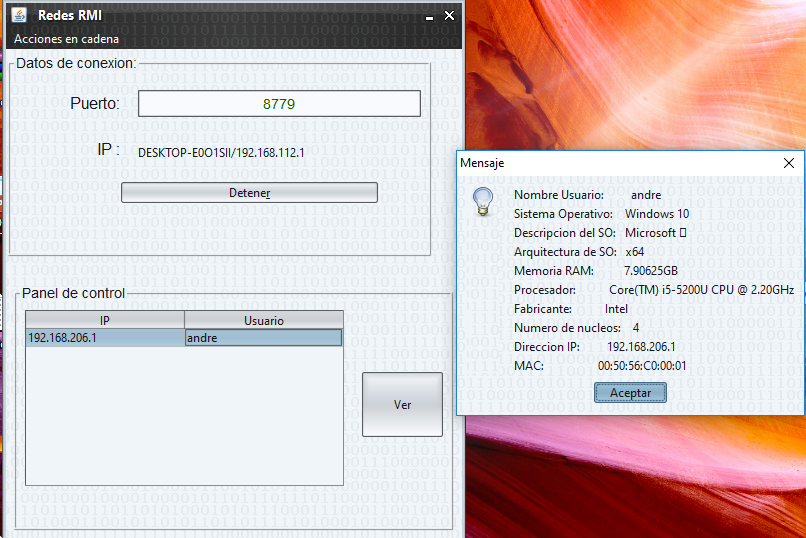
Lo que debemos hacer es iniciar el servidor pasándole el puerto por el cual vamos a trabajar, después de iniciarlo se inician los cliente y se les coloca el puerto y la ip del servidor, al conectarse inmediatamente el cliente nos saldrá en la tabla la mac del cliente y el nombre.



Al seleccionar una fila de la tabla y presionar el botón ver nos carga un panel con la información de ese equipo, el método que se utilizó para comparar y seleccionar el panel es el siguiente



Donde lo primero que hacemos es iterar sobre la lista de clientes, y comparar la IP de estos con la que se seleccionó en la tabla, dando el siguiente resultado:



1. <https://es.wikipedia.org/wiki/Java_Remote_Method_Invocation> [↑](#footnote-ref-1)