|  |  |  |  |
| --- | --- | --- | --- |
| logo_udlap    **EDEI** | **Recuperación de Información**  **Trabajo Práctico III** | | |
| **DEPARTAMENTO** | Computación, Electrónica y Mecatrónica | **MATERIA** | IS – 346 |
| **PROFESOR** | José Luis Zechinelli Martini | **PERIODO** | Otoño 2013 |
| **INTEGRANTES:** Rachid Cesín Gorostieta- 142768, Ulysses Lince Romero- 144088, Jorge Manuel Orozco Prado- 140398 | | | |

**Remote Method Invocation**

1. **Objetivo**

En esta práctica se revisó el procedimiento y las consideraciones que deben revisarse durante la construcción de una aplicación con Java RMI. Se tomó como objeto de estudio una aplicación simplificada de una casa de subastas. A continuación se explica la metodología, los aspectos técnicos de arquitectura utilizados y [blah]

1. **Introducción**

Dentro de nuestro sistema de subastas, el proceso se lleva a cabo de la siguiente manera:

* Un usuario se conecta y puede ofrecer un producto para subastar, estableciendo un precio inicial.
* Por otra parte, el usuario puede revisar el catálogo de productos ofertados, y realizar una puja sobre un producto seleccionado. Esta oferta se puede hacer una o varias veces, siempre y cuando su valor sea mayor al monto actual del producto.
* Al hacer la oferta de un producto inicial, todos los clientes pueden observar la actualización del catálogo de productos.

1. **Arquitectura**

La aplicación está diseñada con una arquitectura **cliente-servidor** para implementar la distribución. Tanto las entidades cliente como servidor están localizadas en la misma máquina, y la comunicación entre ambas se da a través del **middleware RMI** de Java. Cabe señalar que existen situaciones en la que el cliente funciona como servidor y viceversa.

El **servidor** está conformado por las clases Agente.java, Tienda.java; el cliente está formado por la clase Cliente.java, ClienteInterface.java y un applet ClienteApplet.java.

Del lado del servidor, Agente.java es una interfaz que extiende java.rmi.Remote, donde se declaran los métodos que los clientes pueden invocar. Tienda.java es la clase que implementa Agente.java y los métodos remotos que invocarán los clientes; además, registra su stub con el servidor de nombres para que los clientes lo puedan recuperar y crea un servant.

Del lado del **cliente**, ClienteInterface.java es una interfaz que extiende java.rmi.Remote y declara algunos de los métodos que implementa Cliente.java, que por su parte es la clase que recupera el stub del servidor de nombres e invoca métodos del servidor para llevar a cabo sus funciones. ClienteApplet.java es un applet que sirve como front end para la comunicación con el usuario, el input y output de datos.

1. **Funciones**

Para llevar a cabo las funciones de la subasta, el **servidor** se encarga de mantener el catálogo de productos, usuarios y ofertas en estructuras de datos, tablas de Hash en esta implementación, así como proveer a los clientes de un conjunto de métodos para la manipulación de las mismas; por ejemplo, métodos para registrar usuarios, agregar productos a la venta, agregar ofertas y obtener el catálogo. El cliente, por su parte, interactúa con el usuario y hace las llamadas necesarias a los **servants**, para llevar a cabo las siguientes funciones:

* Registro
* Desconexión
* Poner productos a la venta, especificando su nombre y precio
* Obtener una lista del catálogo de productos a la venta
* Hacer una oferta sobre un producto existente
* Actualización automática de la lista de productos cuando un cliente pone un nuevo producto a la venta.

1. **Manejo de errores**

En cuanto al manejo de errores, la tienda hace tres intentos para conectar con el cliente. Si no lo logra hacer después de dichos intentos, los borra. Para evitar productos repetidos, se implementó una tabla de hash. Así nos aseguramos que no exista un mismo producto dos veces. Por otra parte, todo método es síncrono, para que no haya problemas de acceso. De esta manera, cuando se realiza una nueva oferta o se ofrece un nuevo producto, se manda un update a todos clientes.

1. **Diagramas**

A continuación se presentan los diagramas de secuencia sobre el funcionamiento del sistema.

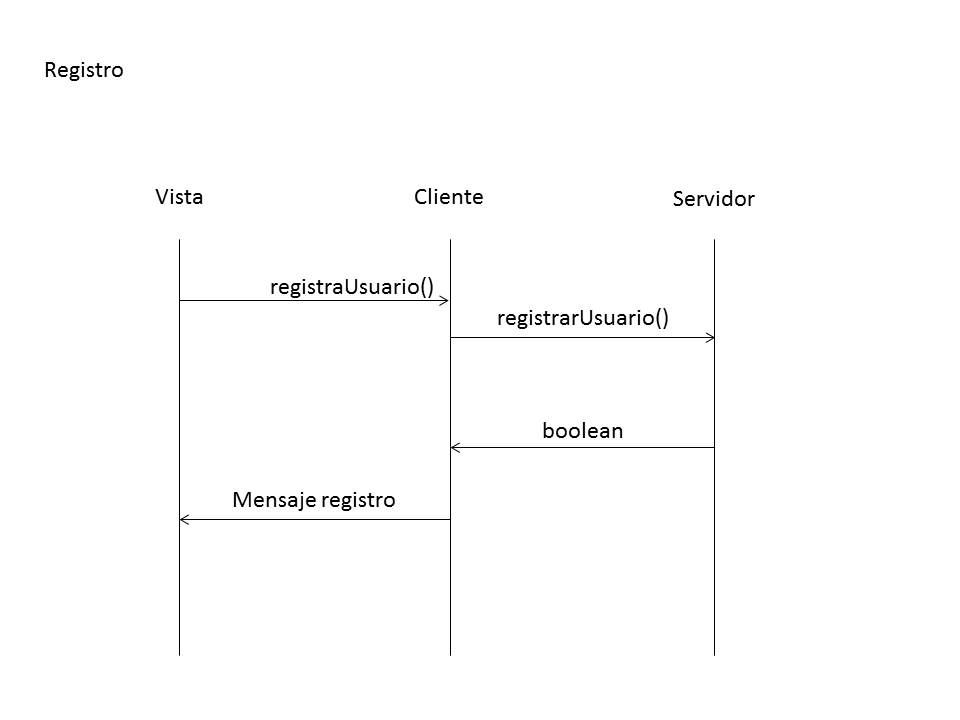


Figura1. Diagrama de Registro

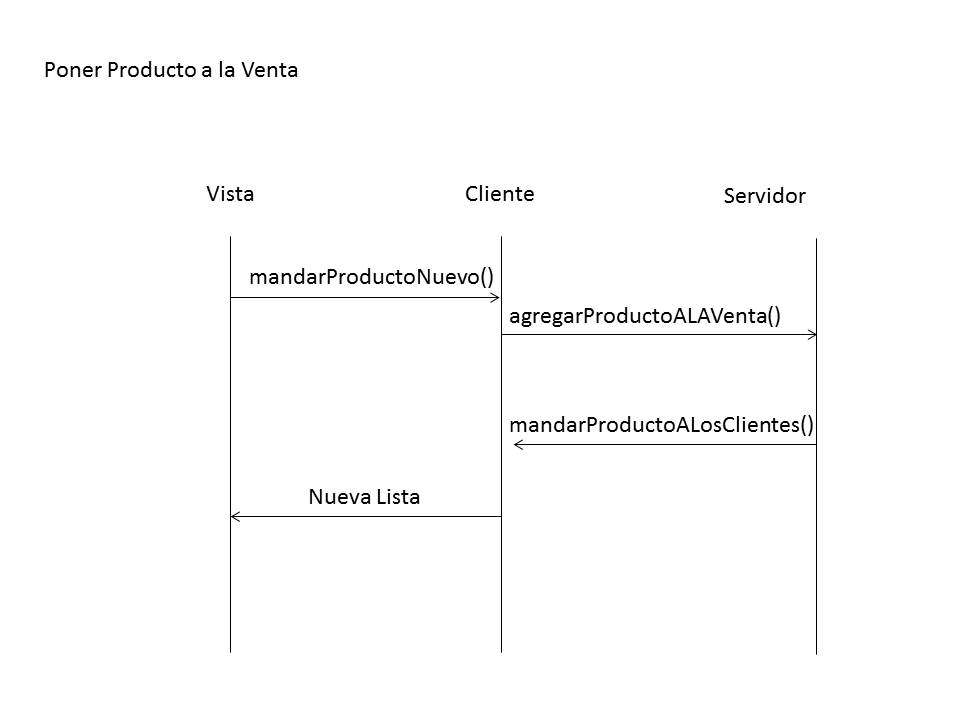
****

Figura2**.** Diagrama de Producto

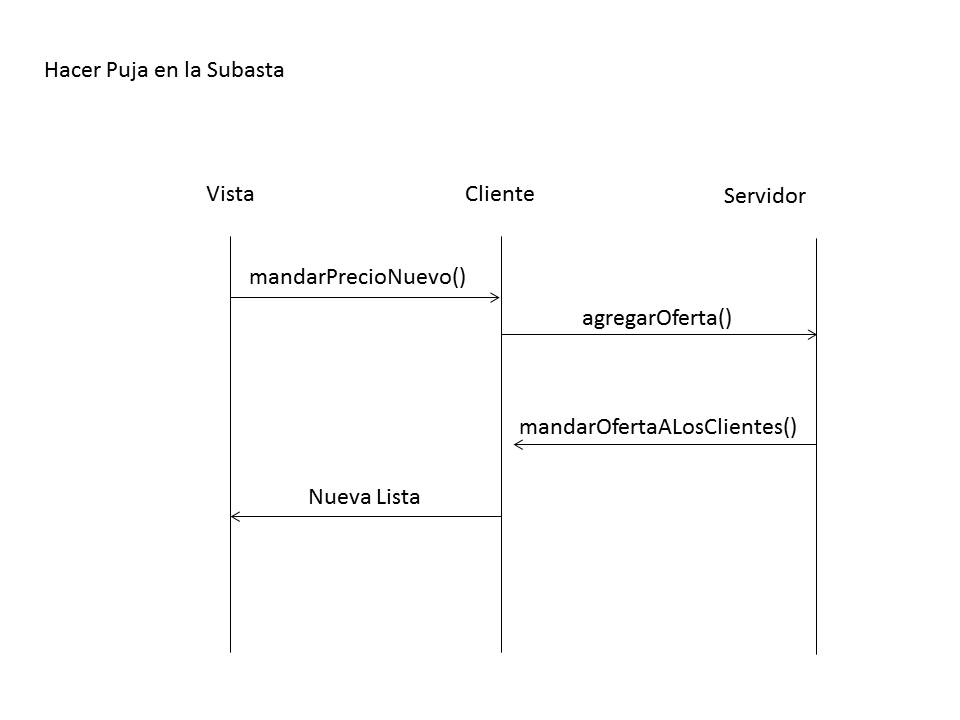
****

Figura3. Diagrama de Puja

1. **Conclusiones**

En conclusión de este proyecto, pudimos apreciar el funcionamiento de RMI, donde se habilita un objeto en la máquina virtual de java para invocar métodos en un objeto sobre otra máquina virtual. (cliente-servidor). La relación entre los diferentes objetos, es definida por una interfaz que extiende la interfaz remota (Subasta en nuestro caso).  
Por otra parte, el stub es utilizado para enviar mensajes (aunado a los parámetros) de la máquina local al objeto remoto e implementa todos los métodos de la interfaz remota.