|  |  |  |  |
| --- | --- | --- | --- |
| **logo_udlap**    **EDEI** | **Sistemas Distribuidos**  **Trabajo Práctico III** | | |
| **DEPARTAMENTO** | Computación, Electrónica y Mecatrónica | **MATERIA** | LIS – 3061 |
| **PROFESOR** | José Luis Zechinelli Martini | **PERIODO** | Primavera 2019 |
|  | | | |
| **Nombre:** | | **ID:** | |

## Práctica RMI

## Introducción

En esta práctica revisaremos el procedimiento y las consideraciones a realizarse durante la construcción de una aplicación distribuida con Java RMI. Como caso de estudio tomaremos el desarrollo una aplicación simplificada de soporte a un sitio para la venta de artículos basado en un modelo de subastas (como lo hace eBay). El escenario típico de utilización de una aplicación de tal índole es el siguiente:

1. Un usuario (jugando el rol de vendedor) se conecta y ofrece un producto, estableciendo un precio inicial y una fecha y hora límite de subasta.
2. Los compradores potenciales se conectan como usuarios corrientes y tienen la posibilidad de visualizar el catálogo de productos disponibles a la compra. El comprador puede entonces seleccionar un producto y realizar una oferta. Cada comprador puede conectarse y ofertar sobre un producto varias veces mientras el periodo de subasta sea aún válido. Notemos que sólo se podrán hacer ofertas que sobrepasen el monto actual (el famoso juego del ¿quién da más?).
3. Al finalizar el periodo de la subasta, el producto es asignado al mejor postor. El vendedor puede entonces verificar los datos del comprador correspondiente, con los que se procederá a contactarlo y así concluir la venta.

Para su operación, tal aplicación debe considerar la administración centralizada de la siguiente información:

* **Datos generales del cliente**: nombre, dirección, correo electrónico, teléfono y un *nickname*.
* **Datos sobre los productos a la venta**: nombre del producto (así como una clave), una pequeña descripción, precio inicial y fecha y hora del cierre de la subasta.
* **Histórico de ofertas a cada artículo**: *nickname* del cliente que hizo la oferta, clave del producto, fecha y hora de la oferta, y monto ofertado.

Notemos que analizar tal aplicación sería complejo, por lo que la versión propuesta es considerablemente más sencilla.

## Versión centralizada

Una práctica recomendada para el desarrollo de este tipo de aplicaciones es la de usar el patrón de diseño MVC (*Model*, *View*, *Controller*) que es ilustrado en Fig. 1:



Fig. 1 Vista conceptual del patrón MVC

Brevemente, este patrón recomienda la separación del código en tres partes: el Modelo, la Vista y el Controlador. El código correspondiente a la parte Vista se centra en los aspectos de interfaz de usuario. El Controlador por su parte recibe los eventos generados a nivel de interfaz y con base en ellos puede dirigir los cambios en el estado de la aplicación (actividad que es delegada al Modelo). Finalmente, la parte Modelo encapsula el estado de la aplicación y generalmente es en este punto donde implementa la lógica aplicativa.

Siguiendo estos principios, la implementación de nuestra aplicación incluye las siguientes clases:

* SubastaVista.java Código de la interfaz de usuario (usando SWING).
* SubastaModelo.java Código aplicativo.
* SubastaControlador.java Implementa el controlador.
* Principal.java Ensambla los objetos que constituyen la aplicación,

además, es el punto de ejecución.

* InformacionProducto.java Información del producto en subasta.
* InformacionOferta.java Información sobre una oferta de compra.

Describamos a continuación la interfaz de usuario implementada por “SubastaVista.java”. La interfaz tiene cuatro zonas (ver Fig. 2):

1. Iniciar una sesión:
   1. Una zona de texto para capturar el nombre de usuario.
   2. Un botón para conectarse.
2. Poner un producto a la venta:
   1. Dos zonas para capturar el nombre del producto y su precio inicial.
   2. Un botón para registrar el producto.
3. Obtener la lista de productos:
   1. Una zona con la lista y un botón para actualizarla.
   2. Una zona de texto con el precio actual del producto seleccionado en la lista.
4. Hacer una oferta:
   1. Una zona de texto para capturar el monto ofrecido.
   2. Un botón para realizar el ofrecimiento.
5. Finalmente, la interfaz incluye un botón para salir del sistema.

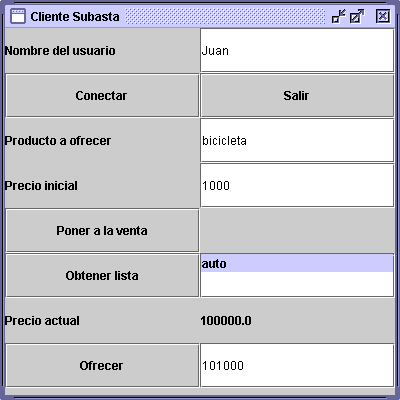


Fig. 2: Interfaz de la aplicación Subasta

Ante la ocurrencia de un evento de interacción (por ejemplo, cuando un botón es presionado), el controlador es notificado. Posteriormente, este evento será interpretado por el controlador quien a su vez recupera alguna información para realizar la llamada del método correspondiente dentro del modelo. Por ejemplo, cuando el usuario presiona el botón “Conectar”, el controlador recupera el valor de la zona de texto que contiene el nombre de usuario y con este valor llama al método “RegistrarUsuario()”.

Para continuar, revise las implementaciones de las clases “SubastaVista.java”, “SubastaModelo.java” y “SubastaControlador.java” (disponibles en la carpeta de código adjunta a este documento). Compile y ejecute el programa. Tome como ejemplo la línea de comando mostrada a continuación:

> javac \*.java

> java Principal

## Versión distribuida

La ventaja principal de haber usado el patrón de diseño MVC, es que la conversión de la aplicación distribuida se hace más sencilla. El código del servidor corresponde con la parte Modelo. Los pasos a seguir son:

* Del lado servidor:
  + Identificar la interfaz para el servidor y adecuar el código para tomar en cuenta tal interfaz.
  + Adaptar el código de clases que serán intercambiadas como tipo de parámetros o de resultados.
  + Agregar el código para la publicación del servidor en el sistema distribuido (nombrado y puesta a disposición) y para inicializar el sistema de seguridad.
* Del lado cliente:
  + Adaptar el código del lado cliente para tomar en cuenta la interfaz del servidor.
  + Agregar el código para la búsqueda y vinculación del cliente con el servidor.
* Ejecución del programa:
  + El servidor debe ejecutarse de la siguiente manera (en dos ventanas separadas):

> rmiregistry

> java –Djava.rmi.server.codebase=URL SubastaModelo

* + El cliente (en una máquina remota, por ejemplo) se ejecuta con el comando:

> java Principal nombre.servidor

## Trabajo a realizar

Las tareas principales del trabajo son:

* Implementar la aplicación de subasta distribuida usando Java RMI.
* Cuando el precio de los productos es modificado, la información no está sincronizada para todos los clientes. Eso quiere decir que al proponer un nuevo precio la vista de los otros clientes no es actualizada. Reflexione cómo se puede resolver el problema y proponga una guía de implementación en RMI.

## Evaluación

La evaluación del trabajo será pondera de la siguiente manera:

* 70% de la calificación: Demostración de la aplicación de subasta con actualización automática de ventanas.
  + Explicación de la arquitectura del sistema.
  + Explicación del funcionamiento interno del sistema.
  + Funciones: agregar y suprimir clientes, ofertas y actualización de ventanas.
* 30% de la calificación: Reporte de la práctica.
  + El reporte debe contener los siguientes elementos: objetivo de la práctica, descripción técnica de la realización de la práctica y lecciones aprendidas con argumentos.
  + El reporte se subirá al sitio del curso en formato PDF. El archivo debe ser nombrado con el primer apellido de los integrantes del equipo.

NOTA: NO se tomarán en cuenta archivos que no respeten estas especificaciones. La ortografía, la buena presentación del documento y la calidad del contenido serán consideradas en la evaluación.