SISTEMA DISTRIBUÍDO DE VOTO

Se crean 4 clases**: VotingServicio, VotingServicioImpl, VotingServidor** que han de estar ubicadas en el equipo servidor, donde se ejecuta la clase **VotingServidor**.
La clase **VotingCliente** se ejecuta en el/los cliente/s.

La función de cada clase es la siguiente:

- **En el servidor**:

1**.- VotingServicio (Interfaz)**: **define los métodos remotos que el cliente va a invocar**.
2.- **VotingServicioImpl (Implementación del Servicio)**: **implementa la lógica de los métodos definidos en la interfaz VotingServicio**. Se ejecuta en el servidor y **es el objeto remoto que los clientes contactarán a través de RMI**.
3.- **VotingServidor (Servidor de RMI)**: es responsable de:

- **Iniciar el RMIRegistry** (el registro donde se inscriben los objetos remotos).
- **Crear una instancia de VotingServicioImpl**.
- **Registrar esa instancia en el RMIRegistry con un nombre** (por ejemplo, "VotingServicio") **para que los clientes puedan localizarla**.
- **Esperar a que los clientes se conecten y hagan invocaciones a los métodos remotos**.

- **En el cliente:**

4.- VotingCliente: se ejecuta en el/los cliente/s.

- **Buscar el objeto remoto** (registrado en el servidor) en el **RMIRegistry** mediante **Naming.lookup()**.
- **Invocar los métodos remotos** definidos en la interfaz **VotingServicio**.
- Interactuar con el usuario, mostrando un menú y gestionando las respuestas del servidor (por ejemplo, votar, obtener recuentos de votos, eliminar candidatos, etc.).

Para probar el ejercicio podemos ejecutar en un terminal una instancia de VotingServidor, y dos instancias de VotingCliente en otros dos terminales. De esa forma simulamos dos clientes interactuando con el servidor, haciendo operaciones diferentes pero modificando y accediendo a los datos globales.

Descripción de cada clase con más detalle

1.- Interfaz remota (VotingServicio)
La clase VotingServicio extiende la interfaz **java.rmi.Remote** y define los métodos que pueden ser invocados remotamente por los clientes. En RMI, es necesario tener una interfaz que declare las operaciones remotas, para que los clientes puedan invocarlas sin conocer la implementación concreta.
Se definen 6 métodos:

1. **int vote(String candidate) throws RemoteException; // Votar por candidato**
2. **int getCount(String candidate) throws RemoteException; // Obtener número de votos de un candidato**
3. **List<String> getAllCandidates() throws RemoteException; // Obtener todos los candidatos con votos**
4. **void removeVotes() throws RemoteException; // Eliminar todos los votos, manteniendo candidatos**
5. **void removeVotesAndCandidates() throws RemoteException; // Eliminar todos los votos y candidatos**
6. **Map<String, Integer> getVotingInformation() throws RemoteException; // Obtener toda la información de voto**

Todos los métodos remotos lanzan **RemoteException**, obligatorio en RMI para manejar posibles errores en la comunicación distribuida.

2.- Implementación remota (VotingServicioImpl)
Esta clase realiza la implementación lógica de los métodos definidos en la interfaz remota **VotingServicio** y extiende **UnicastRemoteObject**, lo que la convierte en un objeto remoto que puede recibir llamadas remotas desde clientes distribuidos.
El uso de UnicastRemoteObject permite que el servidor gestione las conexiones de los clientes, manteniendo la transparencia en la invocación de métodos entre clientes y servidores.
La separación entre la interfaz remota y su implementación permite que la lógica del servidor pueda cambiar sin afectar cómo el cliente interactúa con el servidor.

3.- Registro de objetos remotos (VotingServidor)
Inicia el servidor y registra el objeto remoto en el registro RMI (rmiregistry) permitiendo que esté disponible para los clientes. Utiliza **LocateRegistry.createRegistry()** y **Naming.rebind()** para registrar el objeto remoto bajo un nombre conocido.
La separación entre la lógica del servidor (VotingServidor) y la lógica del servicio (implementada en VotingServicioImpl) permite que el servidor se encargue solo de la gestión de los objetos remotos, sin intervenir en cómo se hace.

4.- Cliente (VotingCliente)
La clase VotingCliente actúa como el consumidor de los servicios remotos.
Utiliza **Naming.lookup()** para buscar el objeto remoto registrado en el servidor. Sigue el modelo de cliente-servidor, donde el cliente no necesita conocer los detalles internos del servidor, sólo interactúa con él a través de la interfaz **VotingServicio**.
Invoca métodos remotos como vote() y getAllCandidates() y recibe la información procesada por el servidor.

Resumen general
La interfaz **VotingServicio** actúa como un contrato que *define las operaciones remotas*, mientras que **VotingServicioImpl** se *encarga de la implementación real*, permitiendo el desacoplamiento entre el cliente y la lógica del servidor.
El servidor usa **rmiregistry** para *registrar el objeto remoto*, y el cliente lo busca a través de **Naming.lookup()**.
El cliente invoca métodos remotos de la misma manera que invocaría métodos locales, lo que es uno de los mayores beneficios de RMI.

Los valores primitivos (como el número de votos) se transmiten por valor, mientras que los objetos remotos (como el propio servicio de votación) se transmiten por referencia. Esto permite que el cliente interactúe con el servidor sin la necesidad de manejar directamente los detalles de la serialización.
Todos los métodos remotos lanzan RemoteException, lo que permite capturar errores de red o de transmisión.