Memoria Ejercicio_2

En el desarrollo de este sistema distribuido de gestión de donaciones, se ha implementado un esquema basado en réplicas de servidor. Cada réplica mantiene su propia instancia del servicio, es decir, su propia estructura de datos, y está registrada en un puerto. Aunque estas réplicas funcionan de forma independiente, en determinadas operaciones es necesario que puedan comunicarse entre sí, y para ello se utiliza RMI(Java Remote Method Invocation). Ademas de las 2 réplicas este programa implementa un cliente Java que se comunica con un sistema distribuido de donaciones con servidores replicados, usando Java RMI (Remote Method Invocation).

Interface RMI: La interfaz Donaciones_I define los métodos remotos que pueden ser invocados por los clientes. Esta interfaz incluye métodos para registrar clientes, realizar donaciones, consultar el total donado, consultar la lista de donantes y obtener el top de los donantes.

Servidores (Réplica 1 y Réplica 2): Existen dos servidores que funcionan como réplicas. Cada servidor tiene una memoria donde almacenan sus propios datos, y ambos servidores permiten registrar clientes y realizar donaciones. Los datos (clientes y sus donaciones) se almacenan en una estructura de mapa (Map) en cada servidor. Las operaciones sobre las donaciones (como registrar un cliente o realizar una donación) se realizan en estos mapas locales. Además, cada servidor puede consultar a la otra réplica para verificar si un cliente está registrado en otra réplica.

Cliente: El cliente se conecta a una de las réplicas esta le permite al usuario interactuar con el sistema. L a replica permite registrar un nuevo cliente en la réplica con menos clientes, es decir, aunque el cliente se conecte a la replica 1 si la replica 2 tiene menos usuarios registrados se conectará a la replica 2. Además se podrán realizar realizar donaciones y consultar diversas estadísticas sobre las donaciones y los donantes. El menú del cliente tiene varias opciones que incluyen la realización de donaciones, la consulta del total donado en ambas réplicas, la visualización de la lista total de donantes y la visualización del "Top de donantes" (ordenado por monto de donación).

El sistema permite a los usuarios:

- Registrarse en una réplica del servidor.
- Realizar donaciones.
- Consultar información agregada de las donaciones como pro ejemplo, una lista de los donantes de mayor donante a menor, o total donado.
- Buscar las donaciones realizadas por otros usuarios.

El sistema garantiza que:

- Un cliente solo puede registrarse una vez.
- Una vez registrado, puede acceder al menú completo.
- Si ya está registrado, el sistema detectará que el usuario ya ha sido registrado por lo que el sistema no le permitirá entrar..

Conexión a las réplicas

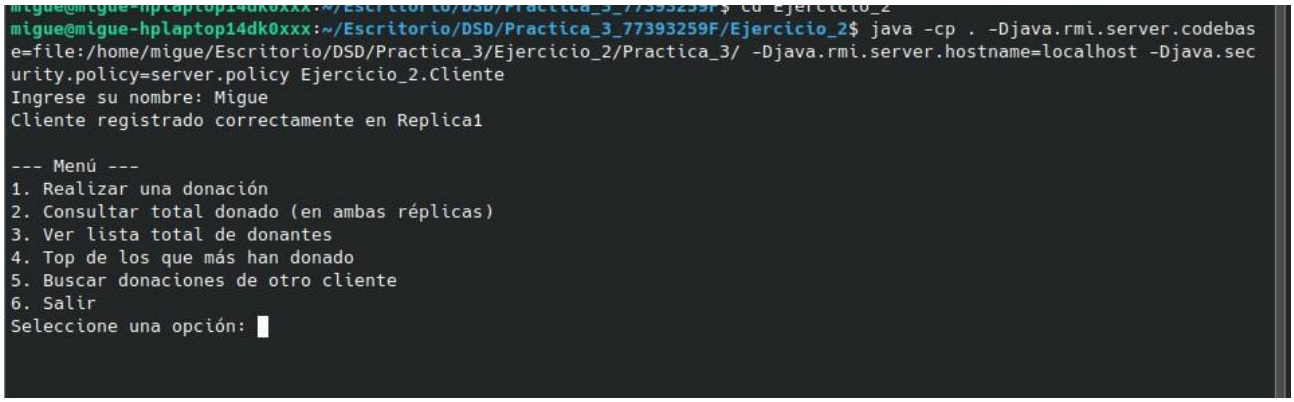
El cliente se conecta a dos réplicas del servidor remoto usando RMI. Cada réplica escucha en un puerto diferente (1100 y 1101) y tiene un nombre registrado (Replica1 y Replica2).

```
Registry registry1 = LocateRegistry.getRegistry("localhost", 1100);
Registry registry2 = LocateRegistry.getRegistry("localhost", 1101);
```

El RMI Registry es un proceso que escucha en un puerto y permite a los clientes buscar objetos remotos por nombre. LocateRegistry.getRegistry() devuelve una referencia al registro RMI remoto que está corriendo en este caso, en "localhost" es decir, en el mismo equipo donde corre el cliente. No se conecta todavía, simplemente crea una referencia que permite hacer búsquedas en ese registro.

Donaciones_I replica1 = (Donaciones_I) registry1.lookup("Replica1");
Donaciones_I replica2 = (Donaciones_I) registry2.lookup("Replica2");

Usa el nombre "Replica1" o "Replica2" para buscar el objeto remoto registrado con ese identificador dentro del registro correspondiente. Estos nombres deben coincidir con los que el servidor usó al hacer el registry.bind("Replica1", replica1) y registry.bind("Replica2", replica2).
El método .lookup() devuelve un objeto de tipo Remote, por lo que hay que hacer un cast al tipo de interfaz que el cliente espera, en este caso Donaciones_I. Se obtienen dos objetos remotos (replica1 y replica2) que permiten al cliente invocar métodos a través de la red como si estuvieran en su propia máquina. Por ejemplo: replica1.consultarTotalDonado() invoca ese método en el servidor.



Conexión entre réplicas

En este sistema, cada réplica necesitará obtener información almacenada en la otra réplica para poder ofrecer diferentes funcionalidades de forma correcta. Por ejemplo; verificar si el cliente ya está registrado en la otra réplica antes de permitir el registro local.

Dado que las réplicas no comparten memoria, esta verificación, así como otras operaciones que necesiten de esta comunicación entre réplicas, ya sea mostrar el total donado en el sistema o la lista de usuarios que han donado, requiere que una réplica acceda a los datos de la otra, lo cual se realiza mediante una invocación remota (RMI)

Desde Replica2, por ejemplo, se puede acceder a Replica1 con el siguiente código:

Registry registry = LocateRegistry.getRegistry("localhost", 1100);

Donaciones_I replica1 = (Donaciones_I) registry.lookup("Replica1");

LocateRegistry.getRegistry("localhost", 1100) se utiliza para obtener una referencia al registro RMI que escucha en el puerto 1100.

registry.lookup("Replica1") recupera el objeto remoto registrado bajo el nombre "Replica1".

Este proceso es idéntico al que realiza el cliente para conectarse al servidor, lo que demuestra que una réplica puede actuar como cliente RMI respecto a la otra. Una vez obtenida la referencia, Replica2 puede invocar métodos definidos en Donaciones_I sobre Replica1

replica1.consultarListaDonantes().contains(nombreCliente);

Selección de réplica y registro automático

```
 Donaciones_I replica;

        if (opcionReplica == 1) {

            Registry registry1 =
LocateRegistry.getRegistry("localhost", 1100);

            replica = (Donaciones_I)
registry1.lookup("Replica1");

        } else {

            Registry registry2 =
LocateRegistry.getRegistry("localhost", 1101);

            replica = (Donaciones_I)
registry2.lookup("Replica2");

        }

        System.out.print("Ingrese su nombre: ");

        String nombreCliente = scanner.nextLine();

  if (!replica.registrarClienteInteligente(nombreCliente)) {

            System.out.println("El cliente ya está registrado en
alguna réplica.");

            return;

        }
```

El cliente se registra automáticamente en la réplica con menos clientes registrados. Esto permite balancear la carga entre réplicas. Si el registro tiene éxito, el cliente puede continuar.

```
System.out.print("Ingrese su nombre: ");
String nombreCliente = scanner.nextLine();
```

Se solicita al usuario que introduzca su nombre. Este nombre será su identificador único en el sistema, que nos permitirá, una vez de almacene este nombre en el sistema, identificar a cada usuario y determinar si ya han sido registrados previamente o no.

```
Registry registry = LocateRegistry.getRegistry("localhost", 1101);
        Donaciones_I replica2 = (Donaciones_I) registry.lookup("Replica2");

        boolean registradoEnEsta = clientesDonaciones.containsKey(nombreCliente);
        boolean registradoEnOtra = replica2.consultarListaDonantes().contains(nombreCliente);

        if (registradoEnEsta || registradoEnOtra) return false;

        int total1 = this.consultarNumeroClientes();
        int total2 = replica2.consultarNumeroClientes();

        if (total1 <= total2) return this.registrarCliente(nombreCliente);
```

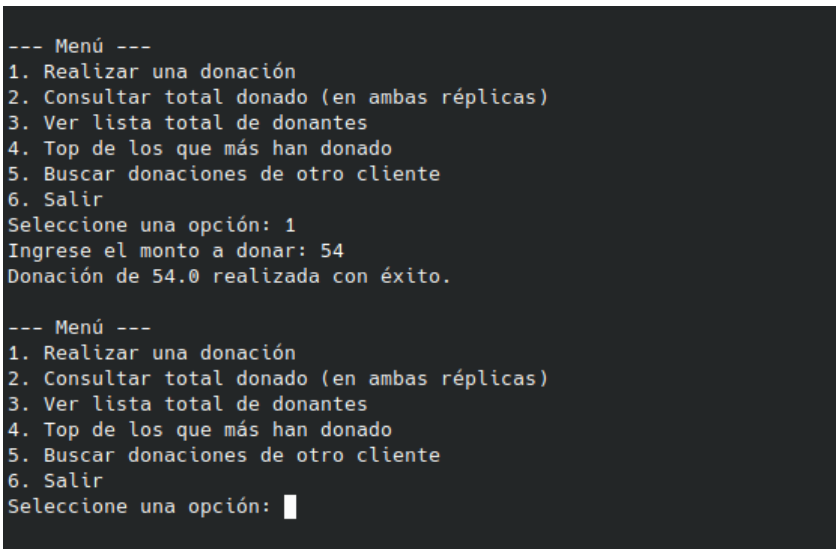
else return replica2.registrarCliente(nombreCliente);

Se verifica si el cliente ya está registrado en alguna de las réplicas. Si es así, no podrá volver a registrarse y el programa termina para evitar duplicación.

Menú interactivo (solo accesible si el usuario fue registrado)

El menú se repite hasta que el usuario seleccione "Salir". Las opciones son:

Opción 1 - Realizar una donación: Permite al usuario ingresar un monto y hacer una donación. La donación se registra solo en la réplica donde el cliente está registrado.



Opción 2 - Consultar total donado: Muestra la suma total de donaciones hechas en ambas réplicas.

Opción 3 - Ver lista total de donantes: Une las listas de donantes de ambas réplicas y las ordena alfabéticamente. Se muestran todos los clientes registrados en el sistema.

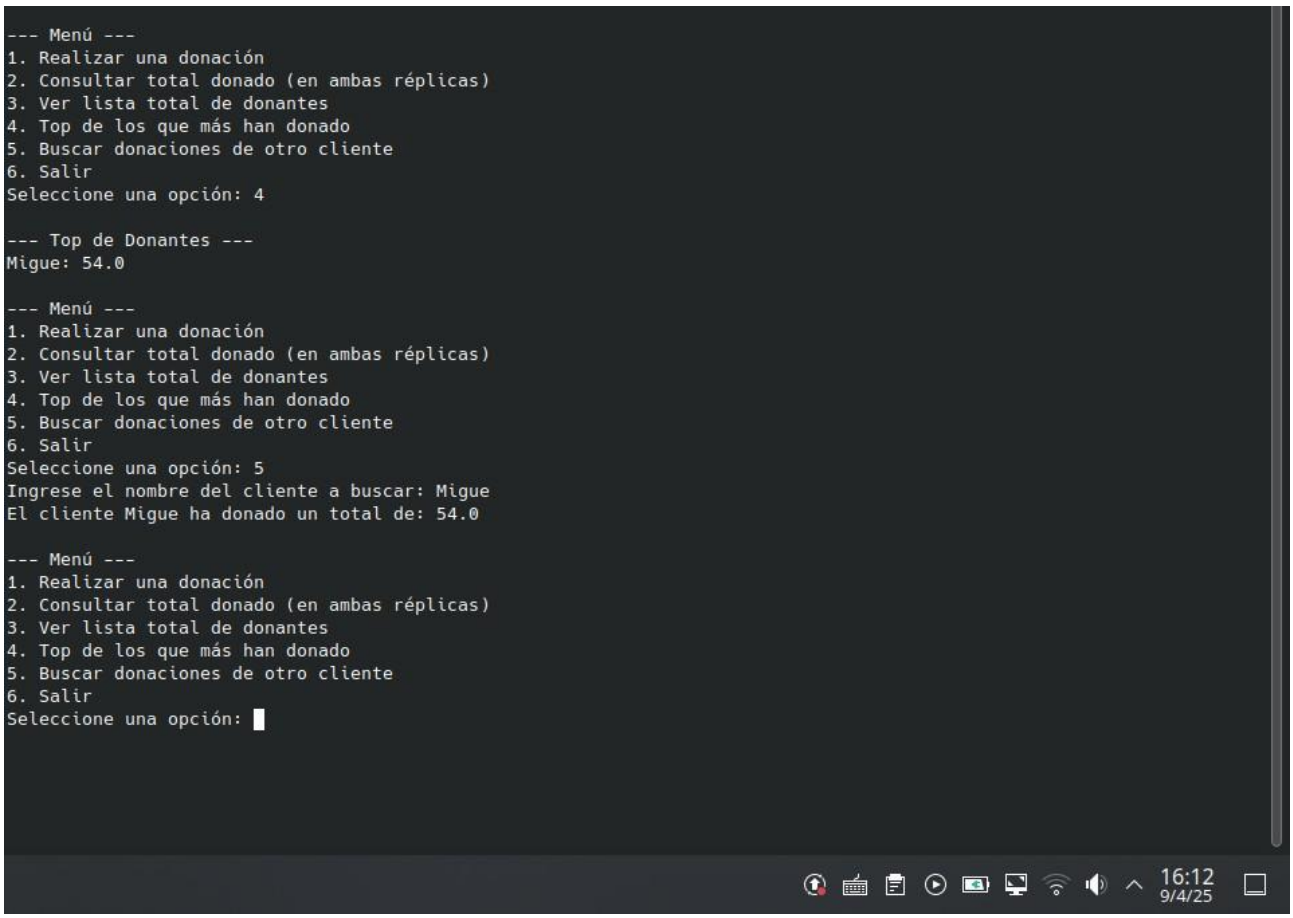
Opción 4 - Top de los que más han donado: Se recopilan todos los donantes y sus donaciones. Se ordenan de mayor a menor según el total donado. Útil para reconocer a los mayores contribuyentes del sistema. Esto se se logra con *List<Map.Entry<String, Double>>* que declara e inicializa una lista llamada listaDonantes, donde cada elemento es una entrada (par clave-valor) de un Map. En este caso:

-String representa el nombre del donante (clave).

-Double representa la cantidad donada (valor).

Así almacenamos tanto el usuario(único en el sistema), como su monto total donado dentro del sistema.

Opción 5 - Buscar donaciones de otro cliente: Permite consultar cuánto ha donado un cliente específico. Se busca en ambas réplicas.



Opción 6 – Salir: Termina el programa de forma segura.

Ejecución del programa

El arbol de directorios del programa es el siguiente:

Compilar todos los ficheros .java con:
javac *.java

Una vez hecho se deben mover a la carpeta Practica_3_77393259F/Ejercicio_2/

Desde la carpeta Practica_3_77393259F/Ejercicio_2 ya podemos proceder:

Iniciar rmiregistry en el puerto **1099**

Ejecutar Replica 2:

java -cp . -Djava.rmi.server.codebase=file:Pwd-local/Practica_3_77393259F/Ejercicio_2/Ejercicio_2/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejercicio_2.ServerReplica2

Ejecutar Replica 1:

java -cp . -Djava.rmi.server.codebase=file:Pwd-loca/Practica_377393259F/Ejercicio_2/Ejercicio_2/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejercicio_2.ServerReplica1

Ejecutar Cliente:
java -cp . -Djava.rmi.server.codebase=file:Pwd-local/Practica_377393259F/Ejercicio_2Ejercicio_2/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejercicio_2.Cliente