

# Práctica 3 - Servidores Replicados

## Desarrollo de Sistemas Distribuidos

**Jose Luis Gallego Peña - DSD2** Todos los ejercicios han sido desarrollados usando el IDE IntelliJ IDEA con Java 13.

### Explicación de la solución

El ejercicio consiste en desarrollar en Java RMI un sistema cliente-servidor en el cual los clientes pueden donar una cantidad de dinero a un servidor. Este sistema tiene las siguientes características:

- Existen varios clientes concurrentes que se conectan a una réplica. Pueden crearse infinitos clientes para distintas réplicas. El cliente dona una cantidad aleatoria de dinero a la replica que se la ha asignado al registrarse. Esta réplica la guarda el cliente.
- Existen varias réplicas del servidor, para ello se instancian N objetos remotos de dos tipos: **Donaciones** y **Replicas**. Estos dos tipos implementan, respectivamente, las interfaces **Donaciones\_I**, para interacciones Cliente-Servidor, y **Replicas\_I** para interacciones Servidor-Servidor. Por tanto, cada réplica, conceptualmente, está compuesta por un objeto de tipo Donacion y un objeto de tipo Replica.

```
// Instanciar n réplicas
for (int i = 0 ; i < numReplicas ; i++) {
    String remoteObjectName = "Replica" + i;

    Replicas replica = new Replicas("localhost", i, numReplicas);    // Servidor-servidor
    Naming.rebind(remoteObjectName, replica);

    Donaciones donaciones = new Donaciones("localhost", replica); // Cliente-servidor
    Naming.rebind(remoteObjectName+"Donaciones", donaciones);

    System.out.println("Replica" + i + " lista");
}
```

Al crear cada Replica se le pasa su identificador y el total de réplicas. Al crear cada Donaciones se le pasa el host y la réplica anteriormente creada asociada a ese objeto remoto de donaciones.

- El cliente usará la interfaz Donacion, que atenderá las peticiones del cliente y llamará a los métodos de la interfaz Replicas. Esta interfaz se comunicará con el resto de réplicas en caso de necesitar datos de ellas.

```
// Interfaz de comunicación cliente-servidor
public interface Donaciones_I extends Remote {
    public int getTotalDonaciones(int idCliente) throws RemoteException, NotBoundException;

    public int registroCliente(int idCliente) throws RemoteException, NotBoundException;

    public void donar(int cantidad, int idCliente) throws RemoteException;
}
```

Esta interfaz define sólo las operaciones que puede hacer el cliente: obtener todas las donaciones del sistema, registrarse o donar.

- Cada objeto Donacion tiene una referencia al objeto Replica que le corresponde.
- Cada réplica almacena:
  - Un map con el id de cada cliente y lo que ha donado ese cliente. Con esto se podrá comprobar si un cliente está registrado y además si ha donado algo, y así poder mostrarle el total de donaciones.

- El identificador de la réplica, un número del 0 a N.
- El número total de réplicas del sistema para saber cuántas hay y contactar con todas.
- El subtotal de dinero donado en esa réplica concreta.

```
// Interfaz de comunicación servidor-servidor
public interface Replicas_I extends Remote {
    public int getTotalDonaciones(int idCliente) throws RemoteException, NotBoundException;

    public int getSubTotalDonaciones() throws RemoteException;

    public int registroCliente(int idCliente) throws RemoteException, NotBoundException;

    public void registrar(int idCliente) throws RemoteException;

    public boolean estaRegistrado(int idCliente) throws RemoteException;

    public int getNumRegistrados() throws RemoteException;

    public void donar(int cantidad, int idCliente) throws RemoteException;
}
```

Esta interfaz permite operaciones entre servidores, obteniendo por ejemplo el subtotal de donaciones, que sólo pueden conocer las réplicas. Es en esta interfaz donde se hace efectivo el registro del cliente y la donación, ya que Donaciones es solo un intermediario.

- El cliente se conectará por defecto a una réplica predeterminada y esta, de manera transparente, gestiona con el resto de réplicas en cual se registrará ese cliente (la que tenga menos clientes registrados). A partir de entonces, el cliente realizará las peticiones a esa réplica en la que esté registrado.

```
public int getTotalDonaciones(int idCliente) throws RemoteException, NotBoundException {
    int total = 0;

    // Obtener total sólo si el cliente está registrado y además ha donado
    if (registrados.containsKey(idCliente) && registrados.get(idCliente) > 0) {
        total = subTotal;
        for (int i = 0 ; i < this.numReplicas ; i++) {
            // Comunicarse con el resto de réplicas
            if (i != this.idReplica) {
                replica = (Replicas_I) registry.lookup("Replica" + i);
                total += replica.getSubTotalDonaciones();
            }
        }
    }

    return total;
}
```

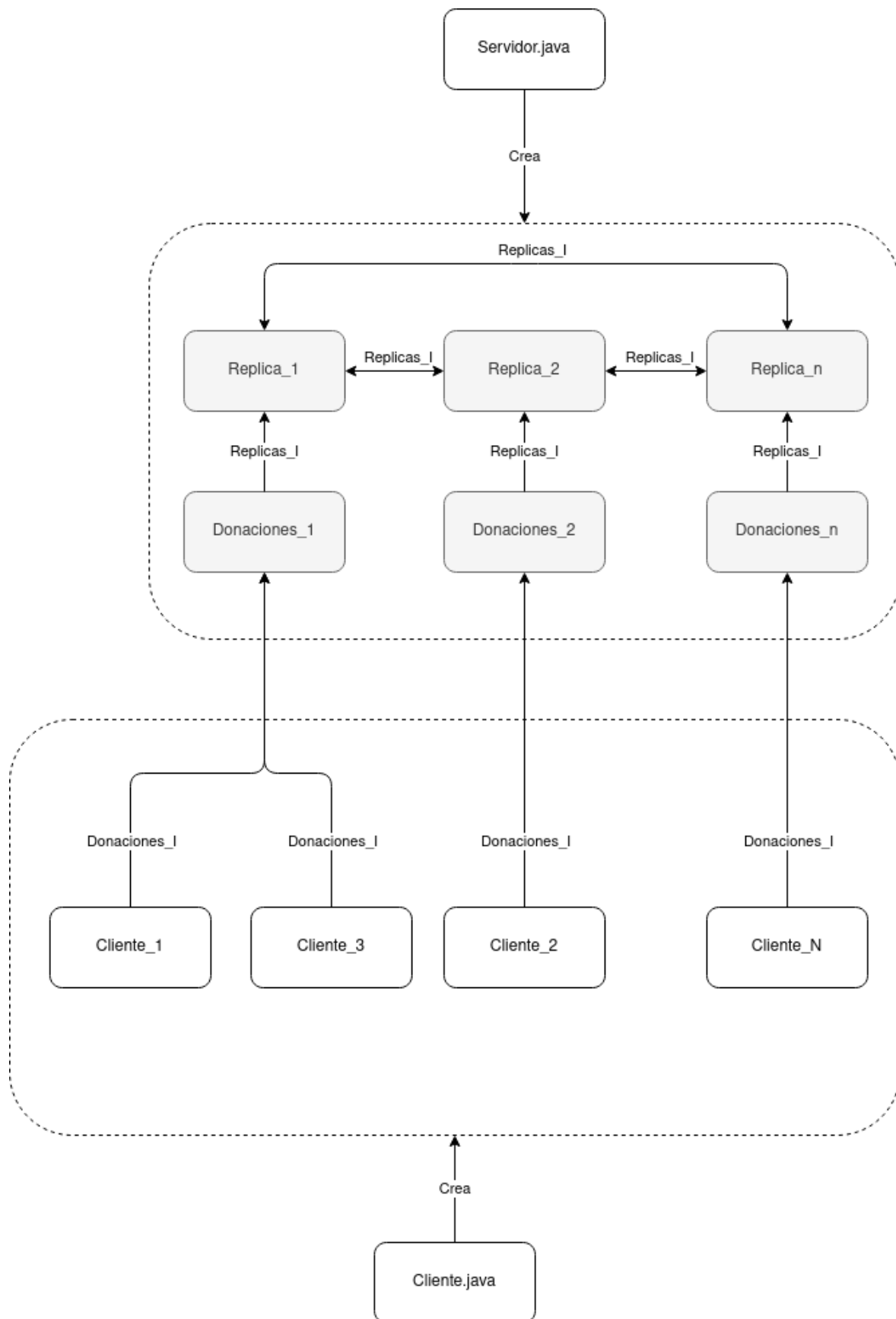
Aquí podemos ver como en la interfaz Replica se conecta con el resto (excepto consigo misma) obteniendo los subtotales de cada réplica y sumandolo a su propio subtotal, para luego devolverlo a la interfaz Donaciones y que esta se la devuelva al cliente.

## Diagrama del sistema

En este diagrama vemos, resumiendo lo anterior, como funciona el sistema. El servidor instancia los distintos objetos Donaciones y Replicas (es decir, N réplicas) y el cliente instancia los distintos clientes, que se conectarán a distintos objetos Donaciones mediante su interfaz, y estos a su vez se conectarán a su objeto

Replicas mediante su interfaz. Cada objeto Replicas se conecta a todas las réplicas menos a sí misma mediante su interfaz.

Los cuadrados blancos indican objetos instanciados y clases, los cuadrados grises son objetos remotos. Las flechas y el nombre que aparecen en ellas indican quién llama a quién y mediante qué interfaz.



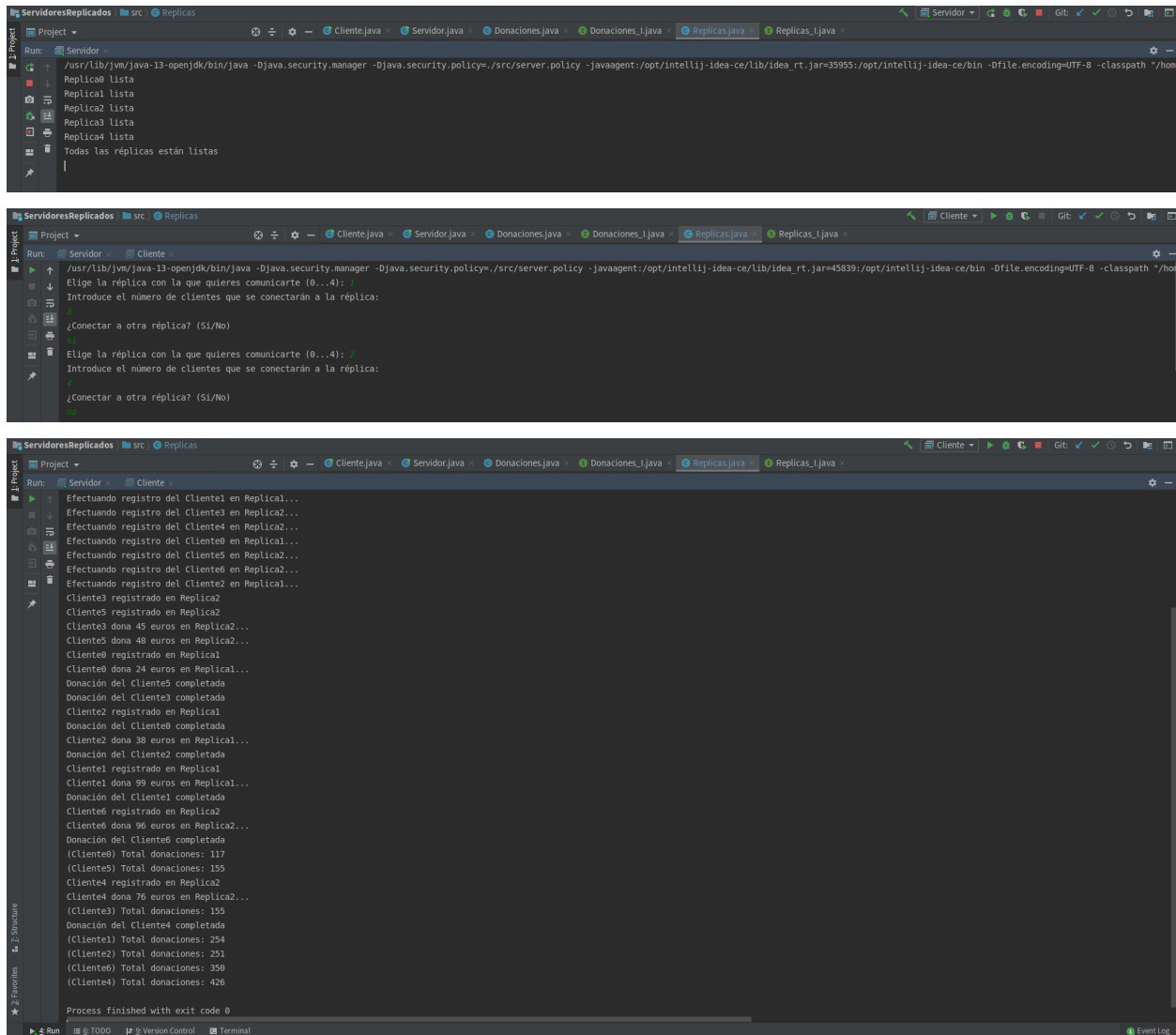
## Ejemplo de ejecución

Por último, vemos un ejemplo de ejecución del sistema.

Por un lado se ejecuta el servidor, que simplemente pone a punto los objetos remotos.

Por otro lado, ejecutamos el Cliente, en el cual primero se nos pregunta a qué réplica conectarnos y luego cuantos clientes se conectarán a esa réplica. Esto se puede repetir infinitamente, el programa pregunta por pantalla si se quieren añadir más clientes a nuevas réplicas. En este caso conectamos 3 clientes a la réplica 1 y 4 clientes a la réplica 2.

Cuando cada cliente termina su donación, muestra el total de donaciones que hay hasta el momento.

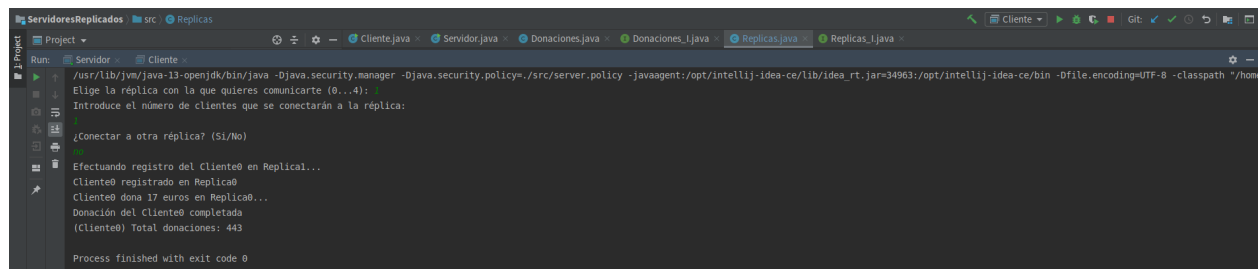


```
Run: Servidor
/usr/lib/jvm/java-13-openjdk/bin/java -Djava.security.manager -Djava.security.policy=/src/server.policy -javaagent:/opt/intellij-idea-ce/lib/idea_rt.jar=35955:/opt/intellij-idea-ce/bin -Dfile.encoding=UTF-8 -classpath "/home/...
Replica0 lista
Replica1 lista
Replica2 lista
Replica3 lista
Replica4 lista
Todas las réplicas están listas

Run: Cliente
/usr/lib/jvm/java-13-openjdk/bin/java -Djava.security.manager -Djava.security.policy=/src/server.policy -javaagent:/opt/intellij-idea-ce/lib/idea_rt.jar=45839:/opt/intellij-idea-ce/bin -Dfile.encoding=UTF-8 -classpath "/home/...
Elige la réplica con la que quieres comunicarte (0...4):
Introduce el número de clientes que se conectarán a la réplica:
¿Conectar a otra réplica? (Si/No)
Elige la réplica con la que quieres comunicarte (0...4):
Introduce el número de clientes que se conectarán a la réplica:
¿Conectar a otra réplica? (Si/No)

Run: Cliente
Efectuando registro del Cliente1 en Replica1...
Efectuando registro del Cliente3 en Replica2...
Efectuando registro del Cliente4 en Replica2...
Efectuando registro del Cliente0 en Replica1...
Efectuando registro del Cliente5 en Replica2...
Efectuando registro del Cliente6 en Replica2...
Efectuando registro del Cliente2 en Replica1...
Cliente3 registrado en Replica2
Cliente5 registrado en Replica2
Cliente3 dona 45 euros en Replica2...
Cliente5 dona 48 euros en Replica2...
Cliente0 registrado en Replica1
Cliente0 dona 24 euros en Replica1...
Donación del Cliente5 completada
Donación del Cliente3 completada
Cliente2 registrado en Replica1
Donación del Cliente0 completada
Cliente2 dona 38 euros en Replica1...
Donación del Cliente2 completada
Cliente1 registrado en Replica1
Cliente1 dona 99 euros en Replica1...
Donación del Cliente1 completada
Cliente6 registrado en Replica2
Cliente6 dona 96 euros en Replica2...
Donación del Cliente6 completada
(Cliente0) Total donaciones: 117
(Cliente5) Total donaciones: 155
Cliente4 registrado en Replica2
Cliente4 dona 76 euros en Replica2...
(Cliente3) Total donaciones: 155
Donación del Cliente4 completada
(Cliente1) Total donaciones: 254
(Cliente2) Total donaciones: 251
(Cliente6) Total donaciones: 358
(Cliente4) Total donaciones: 426
Process finished with exit code 0
```

Si se volviese a llamar de nuevo al programa de los clientes, y por ejemplo el cliente 0 hiciese una nueva donación, esta se sumaría a la que ya tenía en la antigua ejecución del programa de clientes. Esto se guarda siempre que el servidor no se cierre. Llamamos a 1 solo cliente para ver cómo se ha añadido su nueva donación.



```
Run: Servidor - Cliente
/usr/lib/jvm/java-13-openjdk/bin/java -Djava.security.manager -Djava.security.policy=../src/server.policy -javaagent:/opt/intellij-idea-ce/lib/idea_rt.jar=34963:/opt/intellij-idea-ce/bin -Dfile.encoding=UTF-8 -classpath "/hom
Elige la réplica con la que quieres comunicarte (0..4):
Introduce el número de clientes que se conectarán a la réplica:
¿Conectar a otra réplica? (Si/No)
Efectuando registro del Cliente0 en Replicat...
Cliente0 registrado en Replicat0
Cliente0 dona 17 euros en Replicat0...
Donación del Cliente0 completada
(Cliente0) Total donaciones: 443
Process finished with exit code 0
```