Práctica 3. RMI

Desarrollo de Sistemas Distribuidos

Manuel Guerrero Mesías

UNIVERSIDAD
DE GRANADA

Curso 2021-2022

Índice

1. Parte 1

1.1. Ejercicio 1.

Este ejercicio consta de cuatro archivos:

- **server.policy** El cual contienen los permisos de nuestro servidor los cuales especificaremos a la hora de ejecutar el mismo. En este archivo debemos indicarle la ruta donde se alojan los archivos necesarios.
- **Ejemplo-I.java** Es el stub del proyecto, el cual contiene un objeto el cual encapsula un método.
- **Ejemplo.java** Es el servidor del proyecto.
- **Cliente-Ejemplo.java** Es el cliente del proyecto.

Este programa realiza el siguiente funcionamiento:

El servidor se activa y lanza un mensaje Con el cual nos confirma que su funcionamiento es correcto, este mensaje es “Ejemplo bound”. Tras lanzar este mensaje se queda a la espera de recibir alguna solicitud. Aquí es donde entra el cliente el cual posee una instancia local del stub la cual usa para llamar al servidor. El mensaje en concreto, el cual es el “numero de la hebra que va a ejecutarse” se pasa desde el cliente al stub y este lo pasa al servidor.

Estos serian los pasos seguidos en el terminal, primero ejecutar rmiregistry, es opcional poner el numero del puerto pero lo he dejado para evitar errores (se que el 1099 es el puerto por defecto):

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos RMI-20220404/Ejemplo 1$ rmiregistry 1099
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

Lo siguiente seria ejecutar el servidor, al cual le tenemos que indicar donde se encuentran los archivos, la dirección que va a usar y el nombre del archivo con las políticas de uso:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_1$ java -cp . -Djava.rmi.server.codebase=file:/home/manuel/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_1/ -Djav
a.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejemplo
Ejemplo bound
Recibida peticion de proceso: 0
Empezamos a dormir
Terminamos de dormir

Hebra 0
Recibida peticion de proceso: 50

Hebra 50
Recibida peticion de proceso: 3

Hebra 3
```

En la imagen anterior podemos apreciar como ya se han realizado tres invocaciones al servidor, la de la hebra 0, la hebra 50 y la hebra 3, y como su forma de actuar es diferente ya que la hebra 0 va a dormirse siempre antes de terminar su ejecución, mientras que cualquier hebra distinta de la 0 va a ejecutarse sin paradas.

Ahora vamos a ver la ejecución del cliente:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_1$ java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo localhost 0
Buscando el objeto remoto
Invocando el objeto remoto
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_1$ java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo localhost 50
Buscando el objeto remoto
Invocando el objeto remoto
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_1$ java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo localhost 3
Buscando el objeto remoto
Invocando el objeto remoto
```

1.2. Ejercicio 2.

Este ejemplo es similar al anterior en cuanto a los archivos que lo componen, las diferencias entre ambos se encuentran en el contenido de los archivos y por lo tanto el funcionamiento de los mismos. Este programa permite lanzar varias hebras las cuales van a llamar al servidor con su respectivo numero identificador. Al ejecutar el cliente le pasaremos el numero de hebras que queremos ejecutar.

Ahora vamos a comentar los resultados de las ejecuciones del cliente y el servidor, vamos a suponer a partir de ahora que siempre se realizar el rmiregistry antes de ejecutar nada.

El resultado de la ejecución del servidor es el siguiente:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_2$ java -cp . -Djava.rmi.server.codebase=file:/home/manuel/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_2/ -Djav
a.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejemplo
Ejemplo bound

Entra Hebra Cliente 6
Sale Hebra Cliente 6

Entra Hebra Cliente 7
Sale Hebra Cliente 7

Entra Hebra Cliente 5
Sale Hebra Cliente 5

Entra Hebra Cliente 1
Sale Hebra Cliente 1

Entra Hebra Cliente 3
Sale Hebra Cliente 3

Entra Hebra Cliente 8
Sale Hebra Cliente 8

Entra Hebra Cliente 9
Sale Hebra Cliente 9

Entra Hebra Cliente 4
Sale Hebra Cliente 4

Entra Hebra Cliente 2
Sale Hebra Cliente 2

Entra Hebra Cliente 0
Empezamos a dormir
Terminamos de dormir
Sale Hebra Cliente 0
```

En la imagen anterior podemos apreciar que se ha realizado un numero de 5 llamadas al servidor, las cuales se realizan con un orden aleatorio.

Este es el resultado de la ejecución del cliente:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_2$ java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo_Multi_Threaded localhost 10
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
```

1. **¿Qué ocurre con las hebras que acaban en 0?** Las hebras con numero terminado en 0 poseen un estatus especial ya que estas se pondrán a dormir cuando lleguen al servidor.
2. **¿Qué hacen las demás hebras?** El resto de hebras al no poseer dicho estatus especial simplemente harán la misma ejecución que las especiales pero sin dormir.
3. **¿Se entrelazan los mensajes?** Como podemos apreciar en la imagen anterior vemos que las hebras su se entrelazan puesto que no se ejecutan por orden, pero no se entrelazan los mensajes de una hebra con otra.

1.3. Ejercicio 3.

Este ejemplo posee en el servidor una instancia de la interfaz icontador la cual esta contenida en la clase contador, con esto el servidor lo único que hace es rebind con el nombre del objeto que van a buscar los clientes, el cual en este caso sera “mmicontador”. Tras esto el cliente busca el objeto “mmicontador” y se crea una instancia de el a la cual llamara para realizar las peticiones.

El servidor se queda ejecutando esperando tras haber creado el objeto que contiene la interfaz:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_3$ java -cp . -Djava.rmi.server.codebase=file:/home/manuel/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_2/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidor
Servidor RemoteException | MalformedURLExceptiondor preparado
```

El cliente en cambio tras crear su instancia del objeto, empieza a usar este para llamar a las funciones necesarias haciendo lo siguiente: • Poner la variable del objeto a 0. • Incrementar el valor de la variable del objeto hasta 1000 llamando 1000 veces a la función incrementar. • Tras esto se llama a una función para conocer el valor de la variable del objeto la cual debería de valor 1000.

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_3$ java -cp . -Djava.security.policy=server.policy cliente localhost
Poniendo contador a 0
Incrementando...
Media de las RMI realizadas = 0.439 msegs
RMI realizadas = 1000
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P3/Ejemplos_RMI-20220404/Ejemplo_3$ java -cp . -Djava.security.policy=server.policy cliente localhost
Poniendo contador a 0
Incrementando...
Media de las RMI realizadas = 0.246 msegs
RMI realizadas = 1000
```

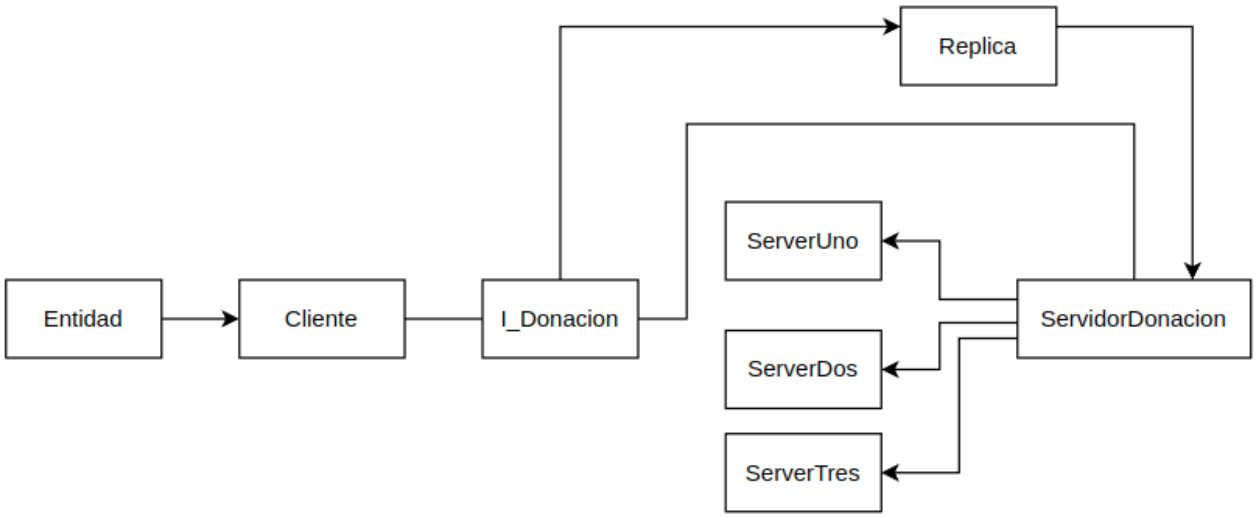
2. Parte 2

2.1. Ejecución del código.

Para la ejecución de este ejercicio se ha creado un script el cual compila y ejecuta los diferentes servidores y clientes para la realización de las comprobaciones pertinentes. Para la utilización de este debemos situarnos donde este el script y ejecutar el siguiente comando: ./script.sh Tras esto se abrirán seis terminales, tres para servidores y tres para los clientes, cada uno de estos clientes se conectara a su respectivo servidor.

Existe una segunda opción del script el cual ejecuta en segundo plano los servidores y tras esto ejecuta tres clientes uno tras otro, este script se ejecuta de la siguiente manera: ./script2.sh

2.2. Explicación del desarrollo.



Sobre este diagrama haré la explicación de este apartado (este diagrama no se ajusta a ningún estándar de programación simplemente a sido creado para ayudar a la explicación del ejercicio).

El código de este ejercicio se ha diseñado de forma que se puedan añadir mas replicas al sistema, por ello la existencia de estructuras de datos como la clase Replica y algunas aplicaciones de la misma, existen para facilitar su comunicación interna.

El cliente se basa en la estructura entidad para su funcionamiento, esta estructura posee el nombre de la entidad que se conecta al servicio de donaciones, así como su estado (si esta o no logueado en alguno de los servidores) y el total que ha donado. Con esta información el cliente se comunica con la interfaz de una de las replicas, la cual es el stub del sistema y permite la conexión con la clase ServidorDonaciones que contiene la implementación de las funciones que va a usar el cliente (como la función de donación que permite donar a la causa), consulta de subtotal donado o consulta del total donado. Estas replicas, que en el ejemplo he llamado ServerUno, ServerDos y ServerTres, se encargan de servir las funciones que necesita el cliente, gracias a esta clase pueden comunicarse como en el caso de a la hora de elegir donde se va a registrar un usuario, el cual se acaba registrando en el servidor con menor numero de entidades registradas.

2.3. Contenido Extra.

Como contenido extra de la practica he implementado:

- Un sistema de registro de sesiones, el cual comprueba que la entidad no este logueada en mas de un servidor al mismo tiempo, por lo cual para loguearse en otro servidor debe cerrar sesión en aquel donde la tiene activa. .
- Un metodo para comprobar el total donado a la causa por todas las entidades de todos los servidores.
- He implementado tres replicas de los servidores en el sistema, en un principio pensé en crear una clase plantilla para que los diferentes servidores se pudieran crear a partir de un mismo archivo y que no tenga que tener cada uno su propio código pero esto me ocasiono diversos problemas y termine dejando tres servidores estáticos, aunque siguiendo la estructura de los servidores dos o tres se pueden crear mas modificando un poco de código de los anteriores y creando nuevos códigos para los nuevos servidores.

2.4. Clases y metodos.

2.4.1. Clase ServidorDonacion (Servidor):

Esta clase compone el funcionamiento de los servidores que se vayan a implementar a partir de esta.

- **String getNombre()**:Devuelve el nombre del servidor.
- **ArrayList<Entidad> getEntidades()**: devuelve el vector de entidades de dicho servidor.
- **Replica getReplica(Replica replica)**: busca entre las diferentes replicas aquella que posee el mismo nombre que la pasada como argumento, si la encuentra restablece sus datos, como hacíamos en modificarEntidad, esto nos permite llamar al servidor correcto en todo momento.
- **boolean isBloqueado()**: Devuelve si el vector esta o no bloqueado.
- **double getDonado()**: Devuelve el total donado en el servidor en cuestión por todas las entidades que han realizado donaciones en el.
- **void setBloqueado(boolean locked)**: establece el estado del servidor a bloqueado.
- **void unlockAll()**: desbloquea todos los servidores.
- **void registrar(String entidad)**:Este método se encarga de registrar una entidad en aquel servidor con menor numero de entidades registradas, si hubiera empate entre varios servidores se registraría en el primero de ellos que hallase en el vector.

- **void donar(String entidad, double cantidad)**: Aumenta la cantidad donada por la entidad tanto en su instancia como en el valor del propio servidor el cual lleva un global del donado en el mismo por todas las entidades.
- **boolean existeEntidad(String entidad)**: Comprueba (mediante el método buscar entidad) si la entidad pasada como argumento existe o no.
- **Entidad buscarEntidad(String entidad)**: Primero busca en este servidor mediante el metodo buscarEntidadAqui, si esta llamada devuelve null buscara en los otros mediante el metodo buscarEntidadAqui de dichos servidores.
- **Entidad buscarEntidadAqui(String entidad)**: Primero comprueba si el vector de entidades esta vacío o no, si no lo esta busca en el vector de entidades del servidor. Si el vector de entidades del servidor esta vacío o no logra encontrar la entidad en este servidor devolvera null, si por otro lado la encuentra devuelve dicha entidad.
- **void modificarEntidad(Entidad entidad)**: Primero comprueba el vector de entidades del servidor esta vacío, si no lo esta comprueba si la entidad pasada como argumento esta en su vector de entidades, si lo esta elimina dicha entidad del vector y añade la pasada como argumento. Con esto se actualizan sus datos para todos los servidores. Si el vector de entidades esta vacío o no se encuentra la entidad en dicho servidor, se buscara en los otros servidores.
- **Entidad iniciarSesion(String entidad)**: Primero comprueba si existe el usuario pasado como argumento, después comprueba si el usuario tiene una sesión activa o no, si no la tiene le activa una sesión. En caso negativo de alguna de las anteriores condiciones no se inicia la sesión y se devuelve null.
- **void cerrarSesion(String entidad)**: Primero comprueba si existe el usuario pasado como argumento, después comprueba si el usuario tiene una sesión activa o no, si la tiene se cierra su sesión. En caso negativo de alguna de las anteriores condiciones se mostrara un mensaje de error.
- **double consultarTotalDonaciones(String entidad)**: devuelve el total donado por la entidad pasada como argumento.
- **double fondoTotal()**: devuelve el total donado por todas las entidades a la causa.
- **double fondoTotalAqui()**: devuelve el total donado por las entidades de este servidor a la causa.

2.4.2. Clase Replica:

Esta clase compone el stub del sistema el cual comunica al cliente con los elementos de interés del servidor.

- **Replica(String nombre)**: Constructor por parámetros.
- **String getNombre()**: Devuelve el nombre de la replica.

- **I**$_{Donacion}getInterfaz() : Devuelvelainterfazdelareplica.$

2.4.3. Clase Entidad:

Esta clase compone la información referente a las entidades y los diferentes métodos para manejar su información. Posee datos como el nombre de la entidad, el total donado por la entidad y su estado de conexión en el sistema.

- **Entidad(String nombre)**: Constructor por parámetros.
- **String getNombre()**: Devuelve el nombre de la entidad.
- **double getTotalDonado()**: Devuelve el total donado por la entidad a la causa.
- **boolean getConect()**: Devuelve la variables conect de la entidad, la cual indica si la entidad esta conectada o no a algún servidor.
- **void setConect(boolean estado)**: establece el valor de la variable conect.
- **void donarDinero(double dinero)**: actualiza la variable totalDonado añadiéndole la cantidad pasada como parámetro a esta variable.

2.4.4. Interfaz I_Donacion:

Esta compone el stub del sistema el cual se encarga de encauzar las peticiones del cliente al sitio correcto donde se encuentra sirviéndose el servidor concreto.

2.4.5. Clase Cliente:

El cliente se encarga de conectar con el stub para disponer de los métodos que va a necesitar del servidor, además da al usuario una interfaz gráfica con la cual puede interactuar para hacer las peticiones necesarias

2.4.6. Clases ServerUno, ServerDos y ServerTres:

Estas clases poseen una instancia de la clase ServidorDonacion la cual enlazan con la interfaz concreta, encargándose así simplemente de instanciar el servidor y procurar todos los sistemas de comunicación necesarios para que el sistema funcione.

2.5. Ejemplo de uso.

Vamos a suponer un ejemplo de ejecución en el cual se va a crear un usuario (U1), el cual se va a registrar, se va a loguear y va a realizar todas las opciones excepto salir y donar, tras esto va a donar un total de 10 y comprobara todas las posibilidades de consulta tras lo cual cerrara sesión.

```
Conectado al ServerTres.
Elige una opcion:
        R - Registrarse
        L - Iniciar sesión
        S - Salir
R
Introduce tu nombre:
U1
Registrado con exito.
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U1
Sesion iniciada.
Buenas U1
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
T
Para consultar lo que has donado antes tienes que realizar una donacion...
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
H
Para consultar lo que has donado en este servidor antes tienes que donar...
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 0.0
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
D
¿Cuánto quieres donar?
10
```

```
Donacion realizada con exito
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
T
Has donado un total de 10.0€.
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
H
Has donado un total de 10.0€ en este servidor.
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 10.0
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
```

Tras esto se va a intentar loguear con un usuario inexistente (U5), después con el usuario que registramos anteriormente (U1) se iniciara sesión y se ejecutaran las mismas operaciones.

```
Conectado al ServerUno.
Elige una opcion:
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U5
El usuario introducido no existe.
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U1
Sesion iniciada.
Buenas U1
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
T
Has donado un total de 10.0€.
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
H
Para consultar lo que has donado en este servidor antes tienes que donar...
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 10.0
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
D
¿Cuánto quieres donar?
10
```

```
Donacion realizada con exito
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
T
Has donado un total de 20.0€.
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
H
Has donado un total de 10.0€ en este servidor.
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 20.0
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
```

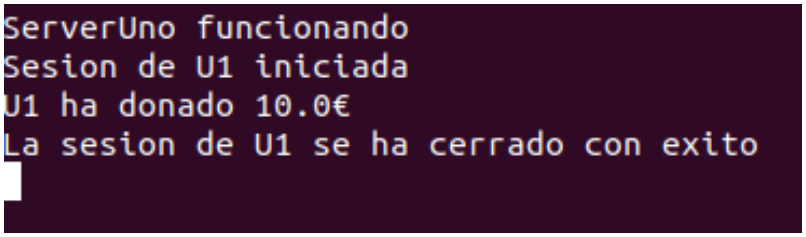
Por ultimo se intentara entra con otro usuario inexistente (U9) y con otro usuario inexistente (U2), tras esto se registrara el anterior usuario (U2) y se realizaran las mismas operaciones que con (U1).

```
Conectado al ServerDos.
Elige una opcion:
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U9
El usuario introducido no existe.
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U2
El usuario introducido no existe.
        R - Registrarse
        L - Iniciar sesión
        S - Salir
R
Introduce tu nombre:
U2
Registrado con exito.
        R - Registrarse
        L - Iniciar sesión
        S - Salir
L
Introduce el nombre de usuario:
U2
Sesion iniciada.
Buenas U2
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
T
Para consultar lo que has donado antes tienes que realizar una donacion...
¿Que quieres hacer?:
        D - Donar
        T - Ver mi total de donaciones
        H - Ver total donado en este servidor
        A - Ver total donado a la causa por todas las entidades
        S - Cerrar Sesion
H
```

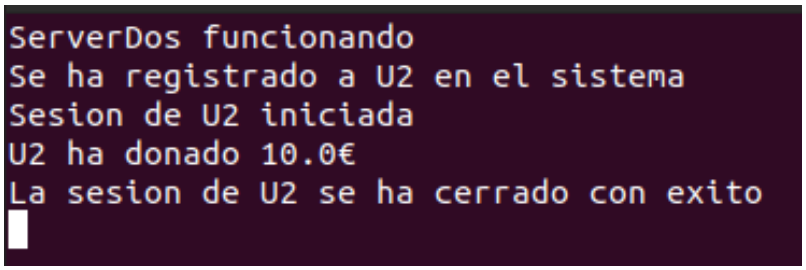
```
Para consultar lo que has donado en este servidor antes tienes que donar...
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 20.0
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
D
¿Cuánto quieres donar?
10
Donacion realizada con exito
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
T
Has donado un total de 10.0€.
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
H
Has donado un total de 10.0€ en este servidor.
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
A
EL total donado a la causa por todas las entidades es 30.0
¿Que quieres hacer?:
         D - Donar
         T - Ver mi total de donaciones
         H - Ver total donado en este servidor
         A - Ver total donado a la causa por todas las entidades
         S - Cerrar Sesion
```

Aquí tenemos los resultados de los diferentes servidores tras la ejecución anterior:

- **ServerUno**



- **ServerDos**



- **ServerTres**