Instituto Politécnico Nacional

Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos

Prof. **Benjamín Cruz Torres**

Práctica No. 4
Pedir cartas en un ambiente distribuido

Grupo: 4CV3

Equipo: 6

Integrantes:

1. Acosta Rosales Jair Sebastián
2. De Jesús López David
3. Galicia Vargas Gerardo
4. Martínez Marcos Luis Eduardo
5. Octaviano Lima Elvia Jaqueline

Fecha: 09 de noviembre de 2018

Práctica 4: Pedir cartas en un ambiente distribuido

Objetivo de la Práctica Que el alumno comprenda la importancia de la programación distribuida y la comunicación dentro de un sistema distribuido.

Tecnologías a aplicar: Sockets, RMI, SOAP, Hilos (threads), POO, Protocolos de comunicación.

Actividades

A partir de la práctica 3, desarrollará una aplicación para repartir cartas baraja francesa a través de un servidor. De acuerdo a los siguientes requerimientos:

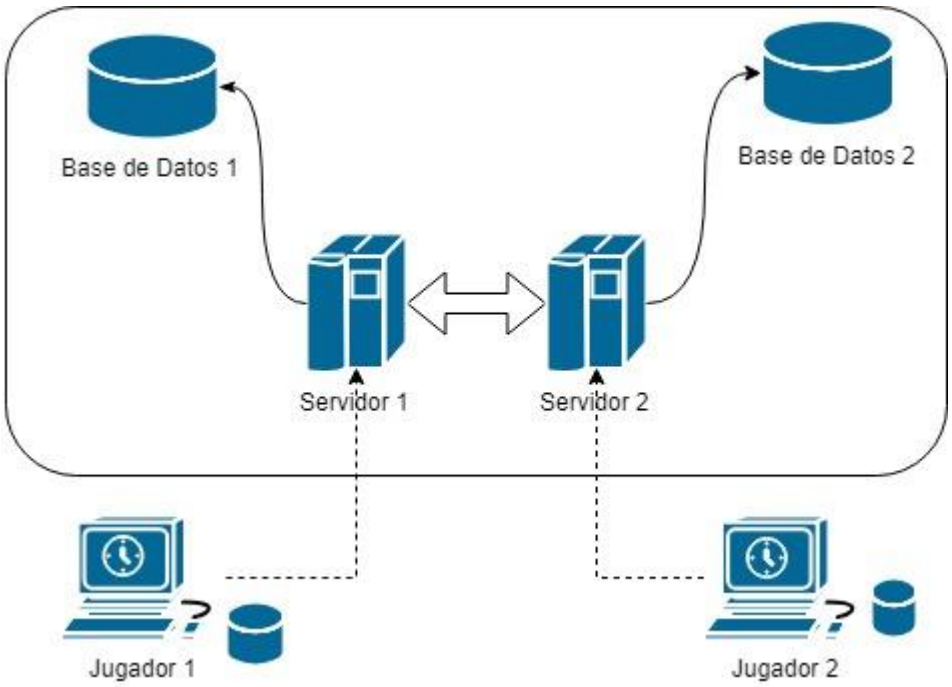


Figura 1. Pedir cartas en un ambiente distribuido

Requerimientos funcionales

- Los servidores son los únicos conectados y con acceso a las bases de datos.
- Dentro de la interfaz de cada servidor habrá un botón de “Reiniciar” y un Canvas para poner una imagen, inicialmente vacío.
- En cada computadora Servidor y Jugador está el reloj de la práctica 1.
- En cada computadora Jugador hay un botón de “Pedir carta”. Éste manda una petición al servidor correspondiente, con la cual dicho servidor envía la información de una carta al azar a ese Jugador.
- La información de petición (IP, hora, carta) se guardará en la base de datos.
- El botón “Reiniciar”, permitirá reiniciar la partida del lado de cada servidor.
- La carta elegida se mostrará (en forma de imagen) solamente en la interfaz gráfica del coordinador. En el cliente se mostrará solamente en formato de texto.
- Cuando termina la partida (se repartieron todas las cartas) se le notificará al usuario si quiere salir o reiniciar una nueva partida.

Requerimientos no funcionales

- El jugador 1 hará peticiones solamente al servidor 1. El jugador 2 hará peticiones solamente al servidor 2.
- Las dos bases de datos mantienen la misma información. Esto es, hay una comunicación constante con entre los dos servidores.
- No se podrán enviar cartas repetidas, a menos que se pulse el botón “Reiniciar” en alguno de los servidores.
- Se recomienda usar la estructura de BD mostrada en la figura 2, por cada servidor.
- El usuario puede modificar cualquier reloj.
- El formato de hora de los relojes es de 24 hrs.
- Al inicio todos los relojes tendrán horas diferentes elegidas al azar.

Investigación

Una de las posibles implementaciones para agregar múltiples clientes a un servidor es mediante hilos. Tanto los clientes como el servidor deberán establecer comunicación a través del mismo puerto, sin embargo, debe agregarse un Receptor, el cual creará un nuevo hilo por cada cliente que se conecte al servidor y tal Receptor se encargará de la gestión de los mensajes entre el cliente y el servidor. En el Receptor se encontrarán dos búferes para gestionar los envíos y recibos del cliente; en un método en específico se definirá qué acción realizar cuando se reciba un mensaje del cliente.

Introducción

En esta práctica se realiza la implementación de Invocación de Métodos Remota (Remote Method Invocation), cuya funcionalidad es la de permitir la colaboración de objetos que están localizados remotamente. Esta tecnología se enmarca en la idea de permitir colaboración entre Objetos Remotos.

El principio de funcionalidad en esta tecnología se podría resumir en los siguientes aspectos

1. Un objeto cliente, crea o completa un requerimiento de datos previamente establecida su estructura en el servidor.
2. El cliente luego prepara el requerimiento que envía a un objeto ubicado en un servidor.
3. El objeto remoto prepara la información requerida (accediendo a bases de datos, otros objetos, etc).
4. Finalmente, el objeto remoto envía la respuesta al cliente

La idea principal o básica de RMI consiste en que, si tenemos acceso a objetos en otras máquinas, podemos llamar a métodos de ese objeto en el otro dispositivo (remoto). RMI maneja los detalles de enviar los parámetros, el objeto remoto debe ser activado para ejecutar el método y los valores deben ser retornados de regreso al equipo que realiza la petición como se muestra en la figura siguiente.

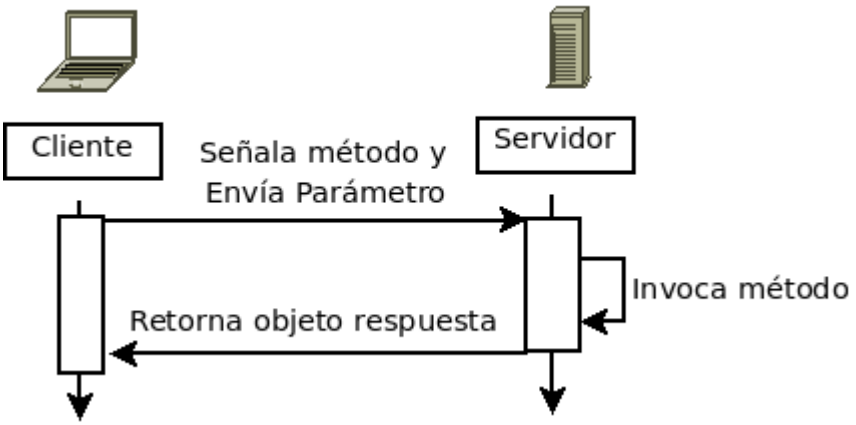


Figura 2. Representación básica del funcionamiento de RMI

Desarrollo de la práctica

Considerando el desarrollo de las anteriores prácticas, se partió del punto en que ya se contaba con la funcionalidad de los relojes y la comunicación entre los clientes y los servidores.

No obstante, se volvió necesario realizar modificaciones en la comunicación entre los objetos del cliente y los del servidor, especialmente para poder agregar a un servidor en el otro y que ambos tuvieran las mismas funcionalidades, pudiendo cada uno realizar la persistencia y las consultas correspondientes en sus datos almacenados localmente.

Para la persistencia se consideró un modelo de base datos con las tablas necesarias para almacenar la información justa sobre las partidas, las cartas y la asignación de estas a los jugadores. El modelo que se realizó es el mostrado a continuación:

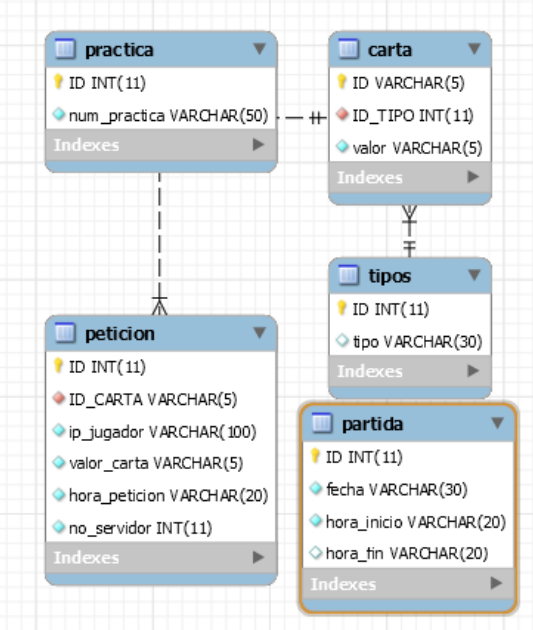


Figura 3. Modelo de base de datos

Los modelos de las bases de datos se poblaron con las rutas de las cartas, cuyas imágenes se encuentran almacenadas en las máquinas donde se ejecutan los servidores. Así, pues el Servidor1 está asociado al modelo *juegocartass1*, mientras que el Sevidor2 se asoció con *juegocartass2*.

Posteriormente, se revisó el funcionamiento del mecanismo RMI, del cual se observó que es provisto por el entorno de ejecución de Java, por lo que una de sus principales limitaciones es que es exclusivo para su utilización con tal entorno de ejecución.

Por otro lado, se observó que consiste en un objeto cliente que realiza un requerimiento de datos, desde uno de sus objetos locales, para después enviarlo a un objeto ubicado en un servidor. El objeto en el servidor realiza las operaciones correspondientes para preparar la información requerida. Finalmente, el objeto en el servidor envía la respuesta al cliente.

Primero se procedió a definir las operaciones de cada cliente y servidor, cuyas firmas se tenían que registrar en la correspondiente interfaz, la cual es la base fundamental de que los objetos mantuvieran una comunicación remota.

Luego, se definieron los detalles de las operaciones, donde los clientes esencialmente solicitan cartas a los servidores (además de una prueba adicional que se incluyó para verificar el funcionamiento, en la que un cliente recibe la consulta que remotamente aplica el servidor sobre su base de datos asociada), mientras que cada servidor agrega al otro y cada uno puede realizar transacciones sobre sus bases de datos asociadas.

La función de reloj en cada cliente y servidor se mantuvo como en las prácticas anteriores, donde cada uno tiene su propio reloj con una hora modificable y es tal hora la que utilizan los servidores para almacenar

Puesto que cada servidor agrega al otro, cada uno debe incluir la dirección IP del otro, no obstante, por el orden en que se ejecutan las operaciones, se debe iniciar primero el Servidor1. Una vez iniciados los servidores, los clientes pueden comenzar a solicitar cartas. Tras la primera solicitud de una carta se inicia una nueva partida. Cada servidor puede reiniciar la partida. Todas las operaciones de partidas y cartas asignadas a jugadores se van registrando en la base de datos.

En la imagen se muestran las interfaces de cada uno de los servidores, con su correspondiente canvas, donde se muestra la imagen de la carta asignada al jugador asociado, y las operaciones que puede realizar. Por su parte, las interfaces de los jugadores muestran el nombre de las cartas recibidas y los botones para sus correspondientes funcionalidades.

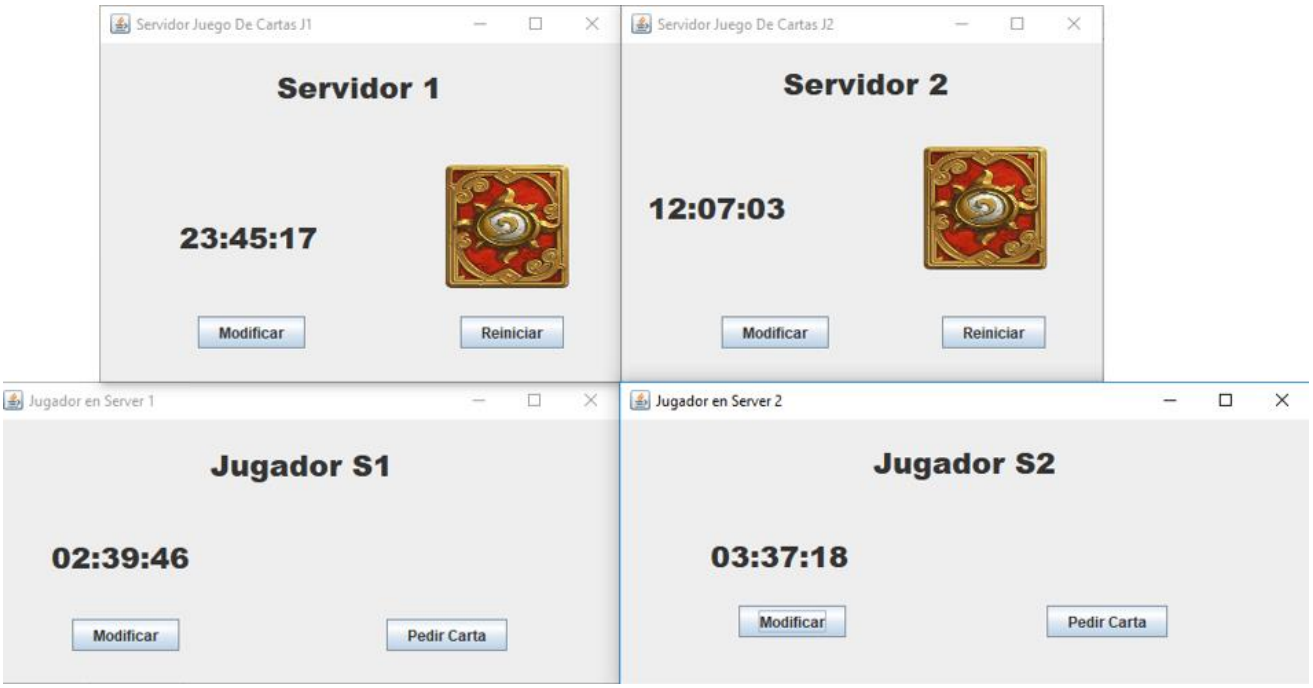
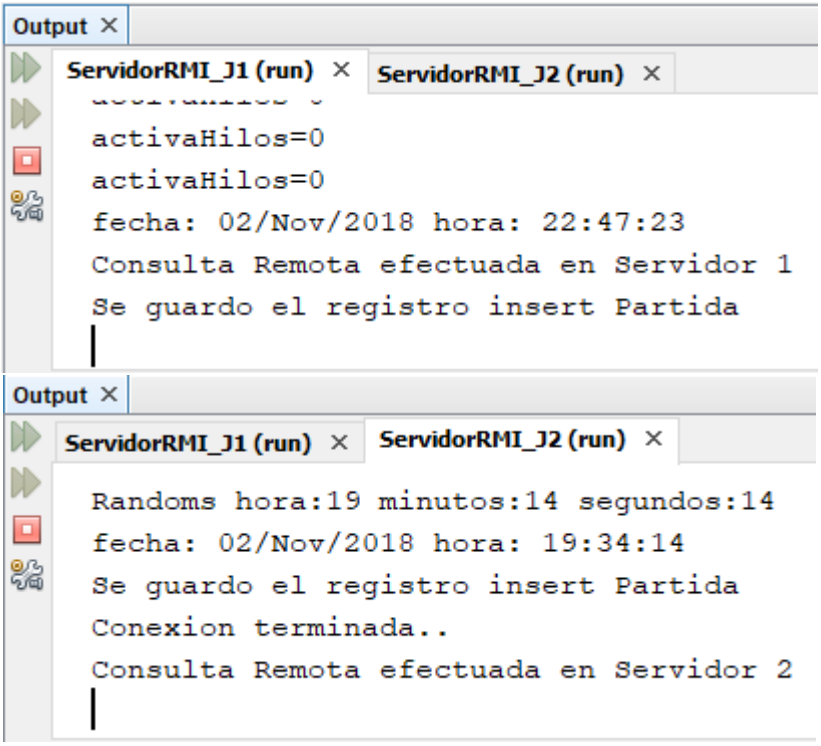


Figura 4. Interfaces de clientes y servidores.

A continuación, se muestran las pruebas realizadas.

Se ejecutan los servidores.

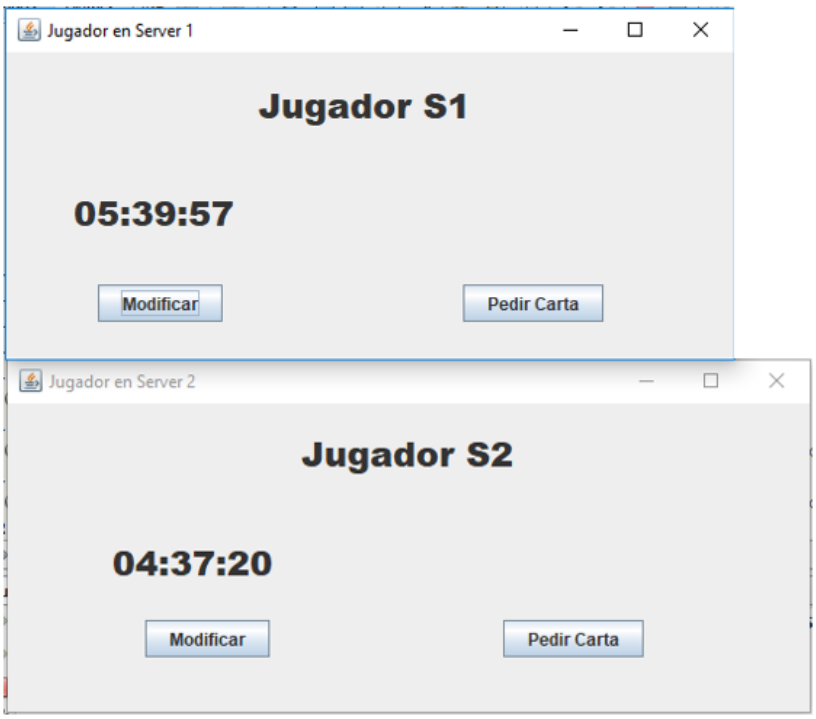
Cada interfaz de servidor se inicializa con un reloj, cuya hora se genera aleatoriamente. En la consola se escriben las conexiones a base de datos y los resultados de las operaciones.



En la base de datos se registra la partida recién iniciada.

ID	fecha	hora_inicio	hora_fin
1	02/Nov/2018	21:20:29	NULL
2	02/Nov/2018	04:37:01	NULL
3	02/Nov/2018	23:30:06	NULL
4	02/Nov/2018	11:51:51	NULL
5	02/Nov/2018	19:34:14	NULL
6	02/Nov/2018	22:47:23	NULL
NULL	NULL	NULL	NULL

Se ejecutan los clientes, cada uno es un jugador, y cada uno se conecta a un servidor correspondiente, por ejemplo, el Jugador S1 se conecta al Servidor 1.



El Jugador1 solicita una carta, el Servidor1 elije aleatoriamente una de las cartas y realiza la correspondiente transacción para guardar la solicitud de carta, el Jugador1 recibe la carta. El Servidor2 obtiene la misma sentencia para persistir la asignación de la carta en su base de datos asociada.

ID	ID_CARTA	ip_jugador	valor_carta	hora_peticion	no_servidor
1	13	us-w10latitt-45/192.168.168.1	3	13:39:00	1
NULL	NULL	NULL	NULL	NULL	NULL

```
ServidorRMI_J1 (run) × | ServidorRMI_J2 (run) × | JugadorRMI_S1 (run) × | JugadorRMI_S2 (run) ×
Consulta Remota efectuada en Servidor 1
Se guardo el registro insert Partida
nombreCarta: 3 de Corazones Clave: 13
idCarta: 13 ipJugador: us-w10latitt-45/192.168.168.1 valorCarta: 3 horaPeticion: 13:39:00
Exito al guardar los datos en la bd(insertPeticion)
```

Se repite la operación, esta vez para el Jugador2 y Servidor2. Esta vez se observan las dos solicitudes de cartas y se ven ambas solicitudes reflejadas en las bases de datos, tanto del Servidor1 como del Servidor2.

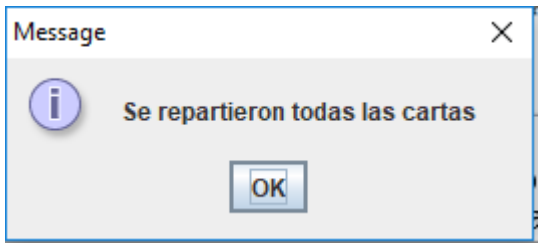
ID	ID_CARTA	ip_jugador	valor_carta	hora_peticion	no_servidor
1	13	us-w10latitt-45/192.168.168.1	3	13:39:00	1
2	14	us-w10latitt-45/192.168.168.1	4	18:59:41	2
NULL	NULL	NULL	NULL	NULL	NULL

```
ServidorRMI_J1 (run) × | ServidorRMI_J2 (run) × | JugadorRMI_S1 (run) × | JugadorRMI_S2 (run) ×
15
nombreCarta: 4 de Corazones Clave: 14
idCarta: 14 ipJugador: us-w10latitt-45/192.168.168.1 valorCarta: 4 horaPeticion: 18:59:41
Conexion terminada..
Exito al guardar los datos en la bd(insertPeticion)
```

Se continúa solicitando cartas hasta que se agotan las disponibles por partida. En la base de datos se observan las peticiones y el servidor que ha asignado cada carta.

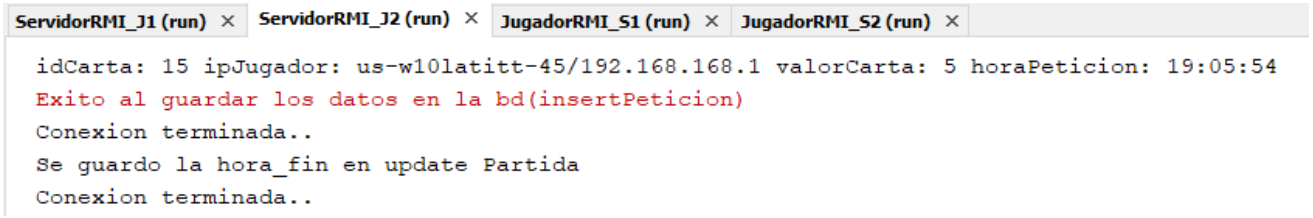
ID	ID_CARTA	ip_jugador	valor_carta	hora_peticion	no_servidor
1	13	us-w10latitt-45/192.168.168.1	3	13:39:00	1
2	14	us-w10latitt-45/192.168.168.1	4	18:59:41	2
3	12	us-w10latitt-45/192.168.168.1	As	19:05:53	2
4	15	us-w10latitt-45/192.168.168.1	5	19:05:54	2

Al agotar las cartas disponibles, se lanza un mensaje.



En la base de datos se registra el fin de partida.

ID	fecha	hora_inicio	hora_fin
1	02/Nov/2018	18:52:58	19:05:55
2	02/Nov/2018	13:37:26	14:00:01



Conclusiones

El mecanismo de RMI, provisto por Java, permite que los objetos cliente realicen peticiones, los objetos servidor preparen y envíen la respuesta, y los objetos cliente reciban la respuesta esperada.

Tal como se menciona, puesto que cada objeto puede hacer uso de los métodos remotos que la interfaz pone a disponibilidad para los demás objetos, este mecanismo ofrece transparencia, pues los métodos se pueden emplear de forma que parece que se están utilizando localmente.

Bibliografía

[1] Tutoriales, W. (2018). Comunicación entre un servidor y múltiples clientes. [online] Webtutoriales.com. Available at: http://www.webtutoriales.com/articulos/comunicacion-entre-un-servidor-y-multiples-clientes [Accessed 27 Oct. 2018].

[2] Gonzalez, A. (2018). RMI: Remote Method Invocation. [online] Profesores.elo.utfsm.cl. Available at: http://profesores.elo.utfsm.cl/~agv/elo330/2s09/lectures/RMI/RMI.html [Accessed 28 Oct. 2018].