|  |  |  |
| --- | --- | --- |
|  | **Instituto Politécnico Nacional**  Escuela Superior de Cómputo |  |
|  | Desarrollo de Sistemas Distribuidos |  |
|  | Prof. **Benjamín Cruz Torres** |  |

**Práctica No. 7  
Sincronización usando relojes lógicos**

Grupo: 4CV3

|  |  |
| --- | --- |
|  | Equipo: 6 |
| Integrantes: | 1. Acosta Rosales Jair Sebastián |
|  |  |
|  |  |
|  |  |

*Fecha:{10/12/2018}*Práctica 7: Replicación en Sistemas Distribuidos

Objetivo de la Práctica Aplicar un modelo de replicación dentro de un sistema distribuido para mantener la transparencia.

Tecnologías a aplicar: Sockets, RMI, SOAP, Hilos (threads), POO, Protocolos de comunicación, Bases de Datos, algoritmos de sincronización, algoritmos de exclusión mutua, replicación.

## Actividades

Desarrollar una aplicación que reparta cartas a los usuarios a través de una Base de Datos, la aplicación estará compuesta por un sistema distribuido con tres equipos. Los tres equipos funcionarán como uno solo y proporcionarán el servicio a los usuarios. Se utilizará un modelo de replicación para trabajar las réplicas.

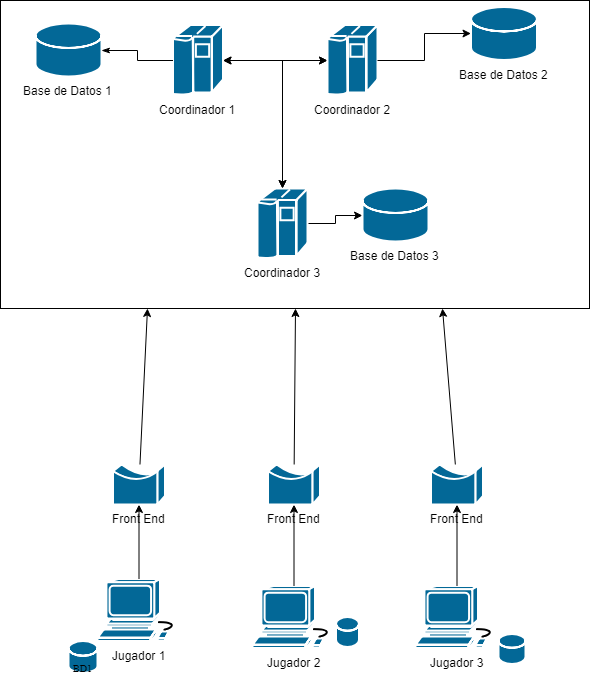


Figura 1. Replicación de Sistemas Distribuidos

## Requerimientos funcionales

* Los servidores son los únicos conectados y con solamente a su propia base de datos.
* Los jugadores se conectarán al sistema principal a través de los Front Ends.
* Los Front Ends elegirán a qué coordinador conectarse.
* Los coordinadores mantendrán un registro de sus actividades en sus BD correspondiente.
* Las Bases de Datos de los coordinadores siempre tendrán la misma información
* Cada vez hay alguna modificación en la BD, se copiarán los cambios a las demás.

## Requerimientos no funcionales

* Los jugadores verán al sistema como una sola computadora.
* Utilice un modelo de replicación para trabajar las réplicas.
* En cada computadora Servidor y Jugador incluirá un reloj lógico o físico.
* La información de petición (IP, hora, carta) se guardará en cada base de datos.
* 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

* Utilizar un algoritmo de sincronización adecuado.

# Introducción

Tomando como base la práctica 6 que permitía consultas remotas de un servidor a otro, está práctica busca implementar alguno de las técnicas de replicación vistas en clase, tales como gossip o la implementada en está práctica **replica primaria,** de modo que nos permita tener en cada una de las bases de datos todo lo ocurrido en el sistema.

Ahora del lado de ambos jugadores se hará uso de lo que se llama Front End, el cual es un programa que permite a cada jugador saber a cual de todos los servidores hacer la petición, esto nos va a permitir ver al sistema como un conjunto y ya no como un servidor dedicado a cada jugador, de este modo el jugador puede ver al conjunto de servidores como uno solo, permitiendo así múltiples opciones para hacer las peticiones.

# Desarrollo de la práctica

**Iniciando server 1**



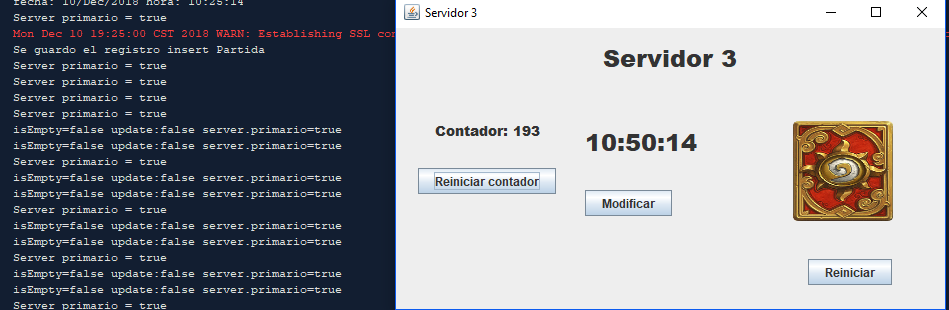
El servidor uno se inicia y queda en espera de que los otros dos servidores inicien también.

**Iniciando server 2**



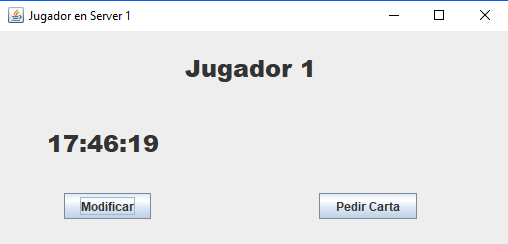
El servidor 2 al igual que el servidor 1 queda en espera de que se conecte el servidor 3, mientras que S1 y S2 ahora están conectados.

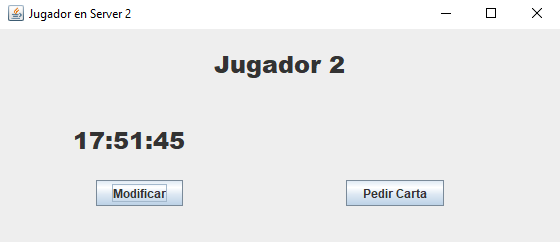
**Iniciando server 3**



Una una vez iniciado el servidor 3 se hace la consulta remota en todas las bases de datos que indican a cada servidor que hay una partida en juego, además cada uno de ellos está en constante espera de alguna petición de un jugador.

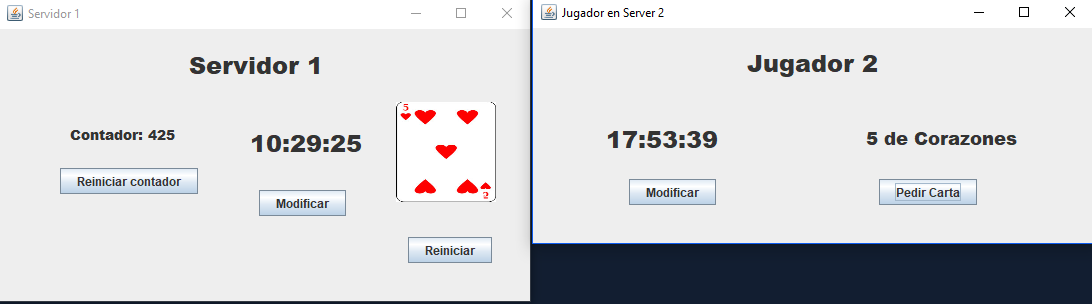
**Iniciando jugadores**



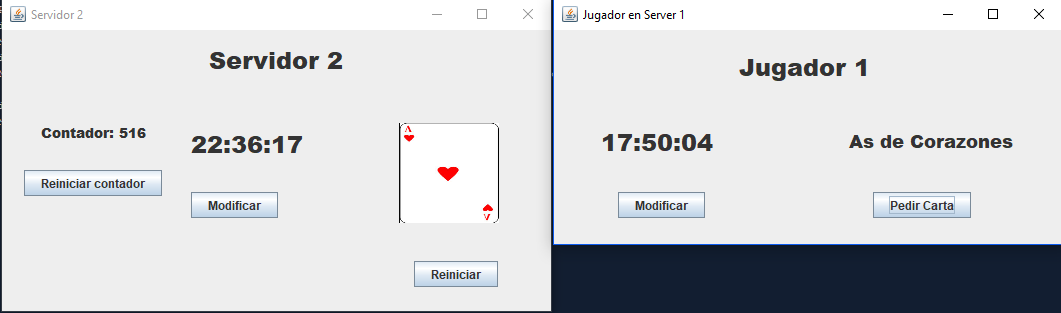


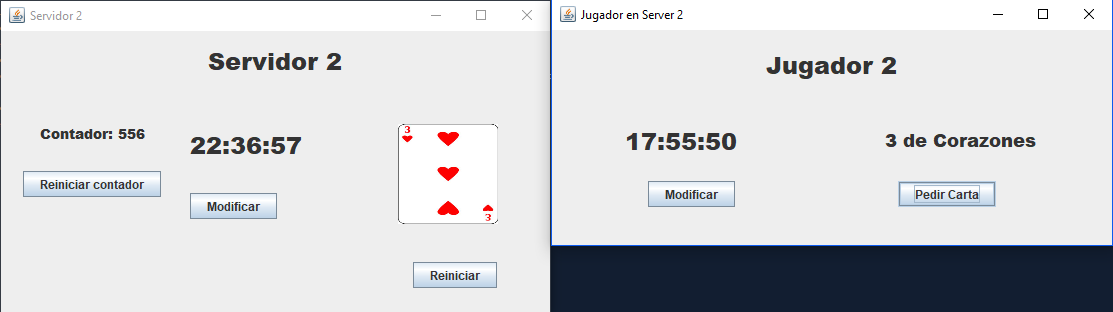
**Haciendo peticiones**



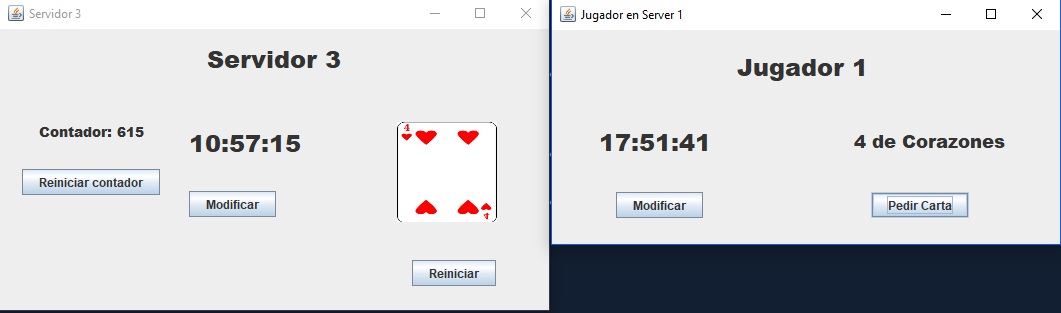


Vemos que la primer opción de los jugadores es el servidor uno, pero si este no estuviera en línea





El servidor 2 es el encargado de atender, y si este también estuviera caído



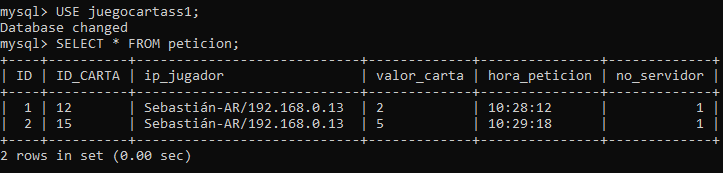
El servidor 3 es el encargado de repartir



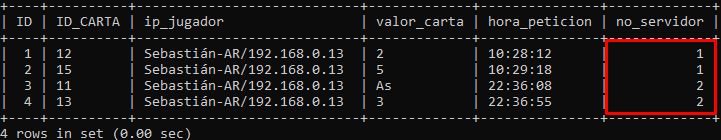
Finalmente acaba de repartir las cartas, dado que aunque este no atendiera las peticiones desde el inicio, internamente le llegaban las actualizaciones de cómo iba sucediendo la partida

**Revisando los registros en las bases de datos**

**La base de datos del servidor 1**

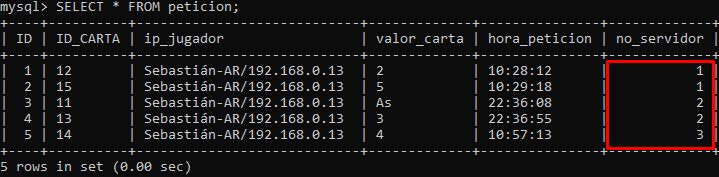


**La base de datos del servidor 2**



# Tiene las consultas del servidor 1 y además las suyas.

**La base de datos del servidor 3**



Contiene todas las consultas que se hicieron en toda la partida

# Conclusiones

Está fue la primer práctica implementando Front End y aunque en un principio no entendía correctamente cómo se empleaba su concepto después logré comprenderlo, aunque cabe aclarar que en está práctica lo importante era la replicación primaria de cada servidor primario a los otros, por lo que el Front End no fue implementado como idealmente debe ser, es decir que este se actualice constantemente para que al hacer la petición no revise todas las ip’s disponibles para conectarse a un servidor.

# Bibliografía

*Comunicación entre un servidor y múltiples clientes*. (s.f.). Webtutoriales.com. Recuperado de: http://www.webtutoriales.com/articulos/comunicacion-entre-un-servidor-y-multiples-clientes

González-J., Agustín. (2009). *Remote Method Invocation (Invocación Remota de Métodos)*. Recuperado de: <http://profesores.elo.utfsm.cl/~agv/elo330/2s09/lectures/RMI/RMI.html>

Alejandro Calderón Mateos, Javier García Blas, David Expósito Singh, Laura Prada Camacho,(2012). JAVA RMI: CALLBACK DE CLIENTE

Recuperado de: http://ocw.uc3m.es/ingenieria-informatica/desarrollo-de-aplicaciones-distribuidas/materiales-de-clase/rmi-callback