Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 40, pp. 62 - 66, Oct 2018, Chihuahua, Chih. México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2018.zip
ISSN 1405-2172

CHAT INTERACTIVO DE COMENTARISTA DE AJEDREZ EN CORBA

Luis Ángel Santamaría Colula, Bárbara Emma Sánchez Rinza
Benemérita Universidad Autónoma de Puebla
Facultad de ciencias de la computación
14 sur y avenida San Claudio
Tel.,52 222 2295500
brinza@hotmail.com

RESUMEN.

El paradigma cliente servidor es el más utilizado para la comunicación de aplicaciones en la red. Para la comunicación de dichas aplicaciones existen herramientas las cuales crean un canal de comunicación que permite enviar y recibir información entre una o varias computadoras. Este artículo tiene como objetivo utilizar la herramienta de CORBA la cual invoca métodos remotos, permitiendo de esta manera la comunicación de servidores de manera distribuida. CORBA permite trabajar diversos lenguajes de programación, de esta manera se crea un estándar entre la comunicación. El objetivo de este trabajo es crear un chat donde se mandan textos e imágenes que hablen de una partida de ajedrez, y el servidor los puede ver.
Se creó este chat un uso particular para no utilizar los chats comerciales, por seguridad y para que este solo se use para fines de comunicación interna entre los comentaristas de la empresa.

Palabras Clave: CORBA, programación, Java, Chat.

ABSTRACT.

The server client paradigm is the most used for the communication of applications in the network. For the communication of said applications there are tools which create a communication channel that allows sending and receiving information between one or more computers. This article aims to use the CORBA tool which invokes remote methods, thus allowing the communication of servers in a distributed manner. CORBA allows different programming languages to work, thus creating a standard between communication. The objective of this practice is to create a chat where texts and images are sent that speak of a game of chess, and the server can see them.
This chat was created a private use to not use the commercial chat, for security and so that it is only used for internal communication purposes among the company's commentators

Keywords: CORBA, Programming, Java, Chat

1. INTRODUCCIÓN

El paradigma *Cliente/Servidor* es quizás el más conocido de los paradigmas para aplicaciones de red. Se usa para describir un modelo de interacción entre dos procesos, que se ejecutan de forma simultánea [1, 9]. Este modelo es una comunicación basada en una serie de preguntas y respuestas, que asegura que, si dos aplicaciones intentan comunicarse, una comienza la ejecución y espera indefinidamente que la otra le responda y luego continua con el proceso. Este paradigma se aplicó en una arquitectura CORBA.
Definiremos primeramente que significa CORBA es un estándar de Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos; es decir, facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.
Las principales características de CORBA son:

- Independencia en el lenguaje de programación y sistema operativo: CORBA fue diseñado para liberar a los ingenieros de las limitaciones en cuanto al diseño del software. Actualmente soporta C, C++, C++11, Lisp, Ruby, Smalltalk, Java, COBOL, PL/I y Python.
- Posibilidad de interacción entre diferentes tecnologías: uno de los principales beneficios de la utilización de CORBA es la posibilidad de normalizar las interfaces entre las diversas tecnologías y poder así combinarlas.
- Transparencia de distribución: ni cliente ni servidor necesitan saber si la aplicación está distribuida o centralizada, pues el sistema se ocupa de todo eso.
- Transparencia de localización: el cliente no necesita saber dónde ejecuta el servicio y el servicio no necesita saber dónde ejecuta el cliente.
- Integración de software existente: se amortiza la inversión previa reutilizando el software con el que se trabaja, incluso con sistemas heredados.
- Activación de objetos: los objetos remotos no tienen por qué estar en memoria permanentemente, y se hace de manera invisible para el cliente.
- la alta capacidad de configuración, libertad de elección de los detalles de transferencia de datos, o la compresión de los datos.

2. ARQUITECTURA DE CORBA

La arquitectura CORBA está orientada a objetos. Los objetos CORBA presentan muchas características de otros sistemas orientados a objetos, incluyendo la herencia de interfaces y el polimorfismo. Lo que hace a CORBA más interesante es que proporciona estas capacidades, incluso cuando es utilizado en lenguajes no orientados a objeto como C o COBOL, aunque CORBA trabaja particularmente bien con los lenguajes orientados a objeto como C++ y Java.

Dentro de las nuevas técnicas y lenguajes de modelado de objetos, cabe destacar la notación estándar UML (*Unified Modeling Language*) [2, 6].

3. INDEPENDENCIA DE LA PLATAFORMA

Un resultado del proceso de clasificación y desclasificación es, que debido a que los parámetros se convierten en la transmisión a un formato independiente de la plataforma, la comunicación entre componentes es independiente de la plataforma [2,7]

. Esto significa que, por ejemplo, un cliente ejecutándose en un sistema Macintosh puede invocar métodos en un servidor ejecutándose en un sistema Unix. Además de la independencia del sistema operativo utilizado, las diferencias de hardware, como puede ser el orden de los bytes más significativos o el tamaño de una palabra, son así mismo irrelevantes, ya que el ORB hace automáticamente la conversión necesaria.

4. EL MODELO DE COMUNICACINES CORBA

Esencialmente, las aplicaciones CORBA se realizan sobre los protocolos derivados de GIOP como IIOP. Estos protocolos van sobre TCP/IP, DCE o cualquier otro protocolo de transporte que utilice la red. Las aplicaciones CORBA no están limitadas a utilizar únicamente uno de estos protocolos; la arquitectura de una aplicación puede ser diseñada para utilizar un puente que interconecte, por ejemplo, componentes de una aplicación basados en DCE con otros basados en IIOP. Es decir, máquinas que suplantan los protocolos de la red de transporte. CORBA es una arquitectura que crea otra capa (la capa del protocolo entre ORBs) que utiliza la capa de transporte subyacente. Esto es también un punto determinante de la interoperabilidad entre aplicaciones CORBA. CORBA no dicta la utilización de una capa de transporte en particular

5. PROTOCOLOS ENTRE ORBS

La especificación CORBA es independiente de los protocolos de transporte; el estándar CORBA especifica el conocido como **GIOP (*General Inter-ORB Protocol*)**. GIOP específica, a alto nivel, un estándar para la comunicación entre varios componentes CORBA ORBs [2, 8, 9].

GIOP, es sólo un protocolo general; el estándar CORBA también determina protocolos adicionales, que especializan GIOP para utilizar un protocolo de transporte en particular. Por ejemplo, los protocolos basados en GIOP existen para TCP/IP y DCE. Adicionalmente, los vendedores pueden definir y utilizar protocolos propietarios para la comunicación entre componentes CORBA.

Los fabricantes tienen que implementar el protocolo IIOP para ser considerados conformes a CORBA, aunque pueden ofrecer además sus protocolos propietarios. Este requerimiento ayuda a asegurar la interoperabilidad entre los productos CORBA de diferentes vendedores pues cada producto conforme a CORBA 2.2 debe ser capaz de hablar el mismo lenguaje [2, 10].

6. DEFINIENDO LA INTERFAZ

Se realizó en el lenguaje Java, a continuación, se detalla la secuencia de pasos que se llevaron para realizar el programa, así como también los comandos que se deben ejecutar para que funcione correctamente.

El primer paso para crear una aplicación CORBA es especificar todos sus objetos y sus interfaces utilizando el Lenguaje de Definición de Interfaz (IDL) de OMG. IDL tiene una sintaxis similar a C ++ y se puede usar para definir módulos, interfaces, estructuras de datos y más. El IDL se puede asignar a una variedad de lenguajes de programación. El siguiente código está escrito en OMG IDL, y describe un objeto CORBA [3, 11] cuyas operaciones son tres las cuales son las de enviar texto, enviar cliente y enviar imagen, devuelve una cadena [4, 5, 9].

7. IMPLEMENTANDO EL SERVIDOR

La clase Server tiene el método main () del servidor, que:

- Crea e inicializa una instancia de ORB
- Obtiene una referencia al POA raíz y activa el POAManager
- Crea una instancia de servidor (la implementación de un objeto Hello de CORBA) y le dice al ORB sobre ello
- Obtiene una referencia de objeto CORBA para un contexto de denominación en el que registrar el nuevo objeto CORBA
- Obtiene el contexto de nomenclatura raíz
- Registra el nuevo objeto en el contexto de nombre bajo el nombre "Hola"
- Espera las invocaciones del nuevo objeto del cliente.

La clase servidor se implementa como se muestra en el diagrama ver figura 1.

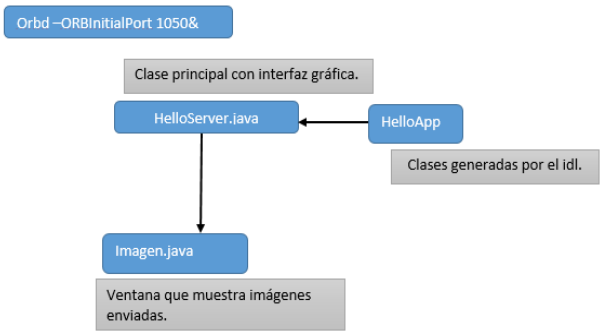

Figura 1. Interacción de la interfaz servidor.

8. IMPLEMENTANDO LA APLICACION DEL CLIENTE

El cliente de aplicación hace lo siguiente:

- Crea e inicializa un orb
- Obtiene una referencia al contexto de nomenclatura raíz
- Busca "Hola" en el contexto de denominación y recibe una referencia a ese objeto CORBA
- Invoca las operaciones que desee realizar del objeto e imprime el resultado.

La interacción de la clase cliente se implementó como se muestra en el siguiente diagrama, ver figura 2.

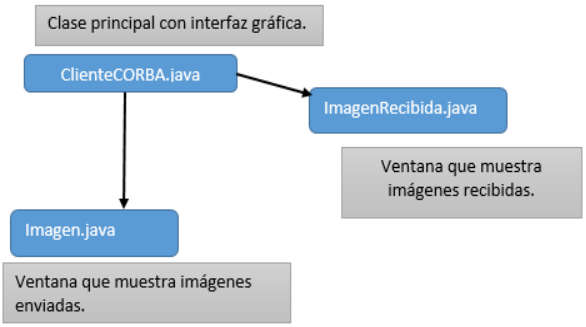

Figura 2. Interacción de la interfaz cliente.

9. RESULTADOS

Para mostrar que la comunicación se encuentra funcionando de manera correcta, se diseñó un servidor el cual escucha las peticiones del cliente [5. 11]. Este servidor muestra en interfaz gráfica las peticiones de los clientes, y cada que un cliente realiza una operación, recibe la información y actualiza los datos, así como también al recibir una imagen muestra una barra de progreso al ser enviada.

La interfaz gráfica del servidor funciona de la siguiente manera:

Primero se corre el ORDB, ver figura 3

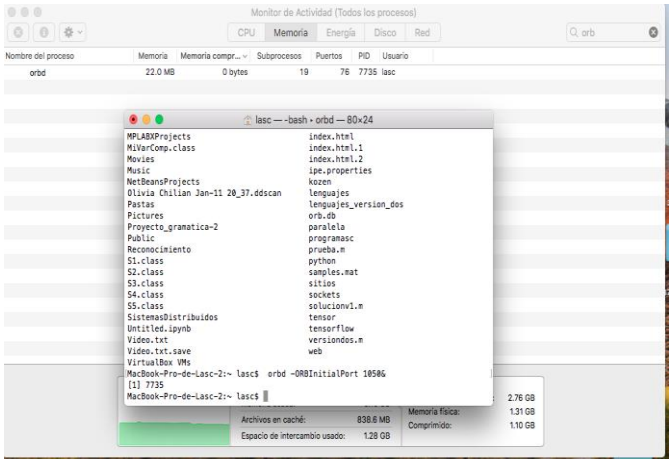

Figura 3. Funcionamiento del ORBD.

Posteriormente se corre el servidor, esté cuando los usuarios se comunican el servidor muestra la información, ver figura 4.

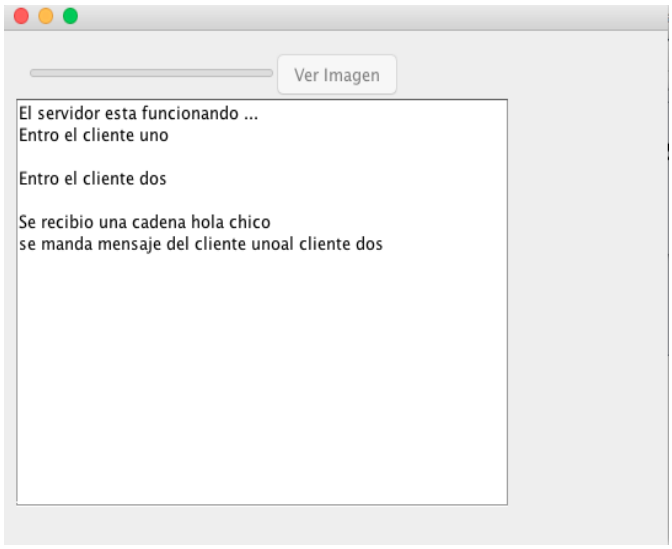

Figura 4. Servidor funcionando.

Al iniciar sesión se habilita el botón para conectar, y los usuarios que estén conectados, como se observa en la figura 5.

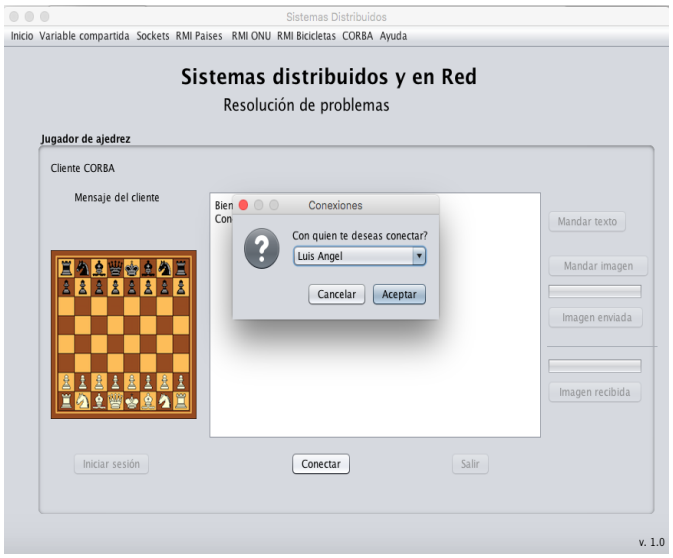

Figura 5. Conexión entre usuarios.

Una vez conectados los usuarios, se pueden mandar mensajes de texto e imágenes, ver figura 6.

Figura 6.-. Envío de mensajes.

Al enviar una imagen el usuario debe seleccionar la imagen en el equipo de cómputo en el que se encuentra. Como se observa en figura 7

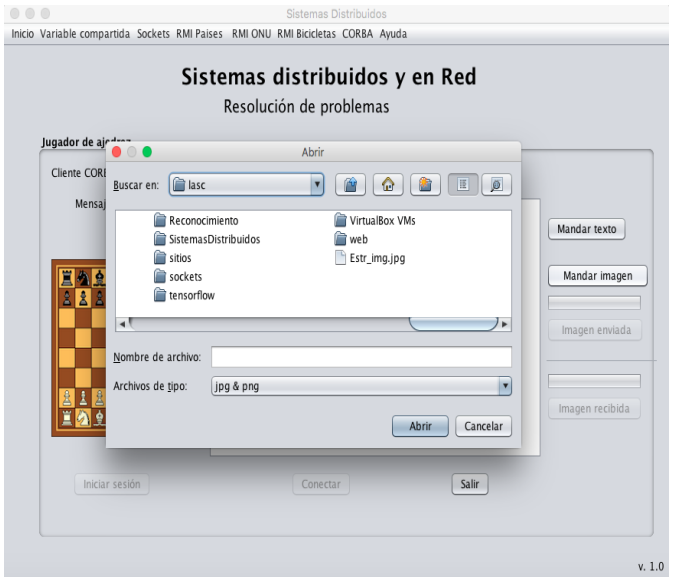

Figura 7. Envió de imágenes.

Al mandar una imagen se carga, y se envía del lado del usuario que manda mensaje, así como el que lo recibe. Ver figura 8.

Figura 8. Carga del envío de una imagen.

Se puede visualizar la imagen enviada en figura 9.

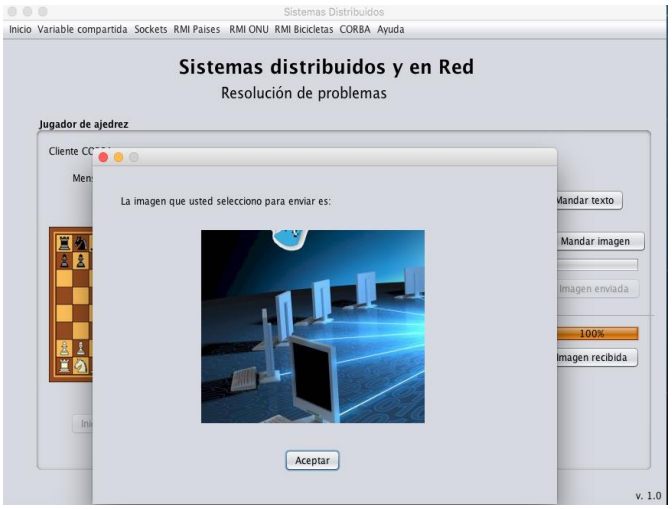

Figura 9. Visualización de una imagen enviada.

10. CONCLUSIONES Y TRABAJOS A FUTURO

Con la utilización de herramientas como Sockets, RMI o CORBA podemos crear proyectos los cuales muestran la comunicación entre un cliente y servidor, así como también ejemplifican cómo es el funcionamiento de la concurrencia en un servidor al atender varios clientes a la vez. La herramienta de CORBA permite la invocación remota de métodos lo cual facilita su utilización, así como también garantizan mayor seguridad para los clientes. Una ventaja que tiene a diferencia de RMI es que esta herramienta no solo funciona en un lenguaje si no es varios. Con este trabajo podemos observar cómo esta herramienta permite una interacción visual facilitando el manejo de los programas.

Este trabajo permite que las empresas puedan tener sus propios chats para fines específicos de trabajos, y no utilizar los chats comerciales

El trabajo a futuro seria meterles seguridad criptográfica a los mensajes, para hacerlos menos vulnerables.

REFERENCIAS

[1] Brookshier, D..”JavaBeans Developer's reference.” New Riders Publishing, 2017

[2] Campione, M.; Walrath, K. “The Java Tutorial Continued: the Rest of the JDK” Addison-Wesley, 2008

[3] Cornell, G.; Horstmann, Cay S. “Core Java” (2ª ed.). Prentice Hall, 2012

[4] Flanagan, D. “JAVA in a Nutshell” (2ª ed.).: O'Reilly&Associates, Inc, 2013

[5] Geary, David M. “Graphic JAVA. Mastering the AWT” (2ª ed.). Prentice Hall. 2015

[6] Jenkins, Michael S. “Abstract Data Types in Java “(1ª ed.). McGraw-Hill. 2011

[7] Horstmann, Cay S.; Cornell, G “Core Java Volume II-Advanced Features.” Prentice Hall: 2016

[8] Orfali, R.; Harkey, D.; Edwards, J."Instant CORBA". John Wiley & Sons, inc.2015

[9] Orfali, R.; Harkey, D.Client/Server Programming with JAVA and CORBA. John Wiley & Sons, inc, 2017.

[10] Sridharan, P. “Advanced Java Networking.”: Prentice Hall 2014

[11] G. Coulouris, J. Dollimore, T. Kindberg. “Sistemas Distribuidos – Conceptos y Diseño” (3ed.) Addison Wesley, 2011