Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 41, pp. 230-234, Oct 2019, Chihuahua, Chih. México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2019.zip
ISSN 1405-2172

IMPLEMENTACIÓN DE UN FORO CHAT PARA EMPRESA

Sección invitada seguridad de la información

Fernando Rojas Ramos, Pedro García Ramírez, Bárbara Emma Sánchez Rinza
Benemérita Universidad Autónoma de Puebla
Facultad de ciencias de la computación
14 sur y avenida San Claudio
Tel. 222 2295500
brinza@hotmail.com

RESUMEN

El trabajo es el siguiente, se implementar un Foro chat también conocido como cibercharla, designa una comunicación escrita realizada de manera instantánea a través de Internet entre dos o más personas, con la capacidad de que varios usuarios se puedan conectar a un mismo servidor y poder comunicarse entre ellos, para discutir dar propuestas sobre un determinado asunto, por ejemplo, en una empresa. La programación se llevó a cabo en Java con la ayuda de la librería Corba que nos brinda el soporte de conexión de varios clientes, con este preámbulo se hicieron varias pruebas de conexión y comunicación para probar su funcionamiento.

Palabras Clave: CORBA, OMG, Java, Chat, IDL

ABSTRACT.

The work is as follows, a chat forum also known as cybercharging will be implemented, it designates a written communication made instantaneously through the Internet between two or more people, with the ability that several users can connect to the same server and be able to communicate with each other, to discuss giving proposals on a particular issue, for example, in a company. The programming was carried out in Java with the help of the Corba library that provides the connection support of several clients, with this preamble several connection and communication tests were made to test its operation.

Keywords: CORBA, programming, Java, Chat.

1. INTRODUCCIÓN

Common Object Request Broker Architecture (CORBA) es un estándar definido por Object Management Group (OMG). Estándar definido por OMG (Grupo de Administración de Objetos). Es un middleware que permite a los componentes de software inter operar en un mismo ambiente sin importar el lenguaje de desarrollo, tipo de red o plataforma [1].

CORBA se adapta fácilmente a los lenguajes de programación como JAVA y C++.

- Los componentes de CORBA son objetos que pueden ser programados en distintos lenguajes y ejecutarse sobre cualquier sistema operativo.

CORBA fue el primer producto propuesto por OMG. Su objetivo es ayudar a reducir la complejidad, disminuir los costos y acelerar la introducción de nuevas aplicaciones informáticas, promoviendo la teoría y la práctica de la tecnología de objetos en los sistemas distribuidos.

Es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas. No obstante, también brinda al programador una tecnología orientada a objetos; las funciones objetos y estos objetos pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa.

CORBA es más que una especificación multiplataforma, también define servicios habitualmente necesarios como seguridad y transacciones. Y así este no es un sistema operativo en sí, en realidad es un middleware.

2. DESARROLLO

En la actualidad existe muchos tipos de chat comerciales, pero realmente con poca seguridad o nula es por eso que este trabajo decide hacer una cibercharla para una determinada empresa donde solo pueden interactuar personal seleccionado de la empresa.

Este sistema de chat simple utilizando Java y CORBA. Lo que se necesita para usar una aplicación que hace uso de CORBA, primero se necesita un ORB (Object Request Broker) el cual se encarga de la comunicación entre los objetos. Se debe tener corriendo para que funcione nuestra aplicación. Más adelante mostraremos como ejecutarlo.
Además, necesitamos un IDL (Interface Definition Language) el cual es un lenguaje que se utiliza para definir las interfaces entre los componentes de una aplicación y es el que soporta la interoperabilidad de la

tecnología. EL OMG definió los tipos estándares para que cualquier lenguaje que se use sepa qué tipo de dato es equivalente en su lenguaje, esto garantiza que podrá ser interpretado [2].

3. DIGRAMA DE FLUJO

Se crea un servidor que sea escuchado por todos los clientes creados, cuando el servidor está listo se establece la comunicación three-way-handshake de acuerdo al protocolo TCP/IP. Los clientes mandan su información y el servidor recibe la información de todos los clientes para que se visualice. El servidor responde a todos los clientes con la información que otros clientes mandaron. Ver figura 1.

Cuando se requiere finalizar la comunicación cada cliente cierra su sesión y se desconecta del servidor.

En la Figura 1 se muestra la lógica de la implementación servidor-cliente.

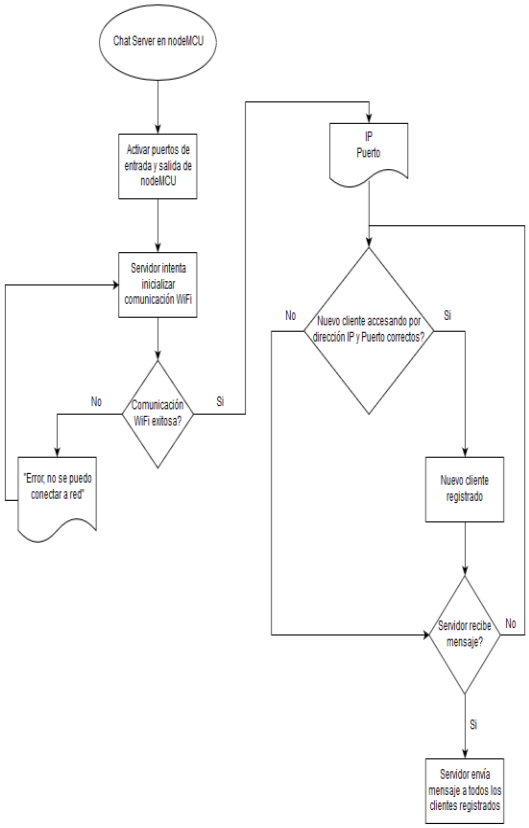

Figura 1.- Lógica cliente – servidor

4- IMPLEMENTACIÓN DEL CÓDIGO.

Las definiciones de la interfaz están abajo. Se muestra a continuación la base del servidor con los clientes que esperan ser escuchados.

```
module chat {
interface ChatListenerI{
void messageReceived (in string message);
};
interface ChatServerI{
void addListener (in ChatListenerI listener); void
sendMessage(in string message);
};
};
```

5. IMPLEMENTACIÓN SERVIDOR.

Aquí está la implementación del servidor. Definimos una clase Server, que heredará de una clase generada automáticamente por el compilador IDL de Allegro OrbLink. Por convención, este chat de clase generado: ChatServerI-servant se encuentra en un paquete con el mismo nombre que el módulo IDL y recibe el nombre de la interfaz con el servidor adjunto.

```
(defclass my-server (chat:ChatServerI-servant)
((listeners :initform nil :accessor get-listeners)))
```

La clase server tiene una ranura, escuchas que contendrán todos los oyentes registrados. Para completar la implementación, necesitamos definir los métodos addListener y sendMessage. Agregar un oyente simplemente empuja al nuevo oyente a la ranura (recuerde, cuando un oyente de Java se crea una instancia, lo primero que hace es llamar a addListener). El método sendMessage toma el mensaje enviado por un oyente y lo transmite a todos los oyentes llamando sucesivamente a messageReceived en cada oyente. El código para mostrar el mensaje en cada escucha es el código de implementación de messageReceived, que, por supuesto, está escrito en Java [3,4].

```
(corba:define-method addListener ((this my- server)
listener)
(push listener (get-listeners this))) (corba:define-
method sendMessage ((this my- server) message)
(dolist (listener (get-listeners this))
(op:messageReceived listener message)))
```

6. IMPLEMENTACION CLIENTE

En el lado de Java, las cosas son un poco más complejas y no mostraremos la implementación completa, solo una parte del método de inicialización que muestra como cada oyente se encuentra

```
private void doConnect(){

orb = ORB.init();
boa = orb.BOA_init();
org.omg.CORBA.Object obj = IorIo.resolve(orb,
filename);// filename is the location of the IOR server
= ChatServerIHelper.narrow(obj); ChatListenerI
listener = new Listener(); boa.obj_is_ready(listener);
server.addListener(listener);
// ... more stuff ... //}
```

El registro en el servidor ver figura 2.

7. SIMULACIÓN.

Antes de comenzar se necesitará tener instalado IntelliJ IDEA [5], se creará un proyecto nuevo, con las siguientes clases ver figura 3.

1.- Se abrirá una ventana de comandos y correr la siguiente línea orbd -ORBInitialPort 1050. Para establecer el puerto de comunicación se aprecia en la Figura 4.
2.- Nos pasamos al director que contiene Conn.idly corre idlj -fall Conn.idl

3.- Construir y ejecutar ConnServeren IntelliJ con arugmentos -ORBInitialPort 1050 - ORBInitialHost localhost o la dirección que se ejecutara como servidor en nuestro caso 172.31.18.65, en la Figura 5 se muestra la respectiva configuración en Intelli.

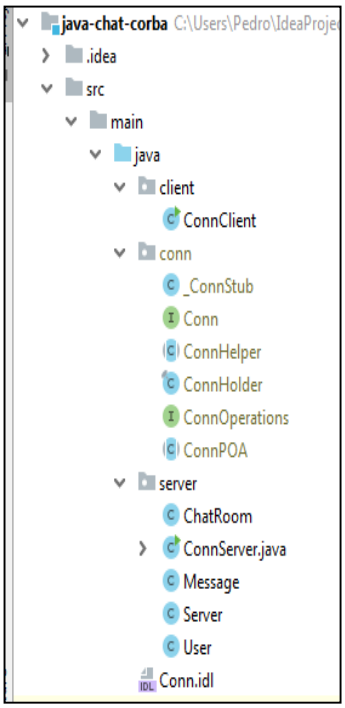

Figura 3. Subcarpetas

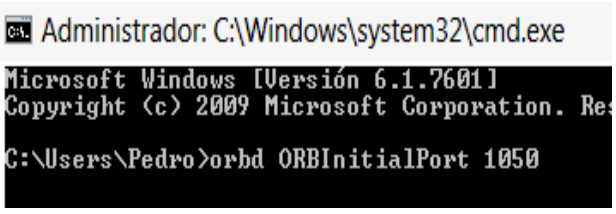

Figura 4. CMD inicializar orbd..

Figura 5. Configuración en IntelliJ.

4.- Construye y ejecuta cualquier número de ConnClienten IntelliJ con argumentos - ORBInitialPort 1050 -ORBInitialHost localhost.

Se puede ejecutar el cliente y el servidor desde IntelliJ como se puede observar en la Figura 6

.

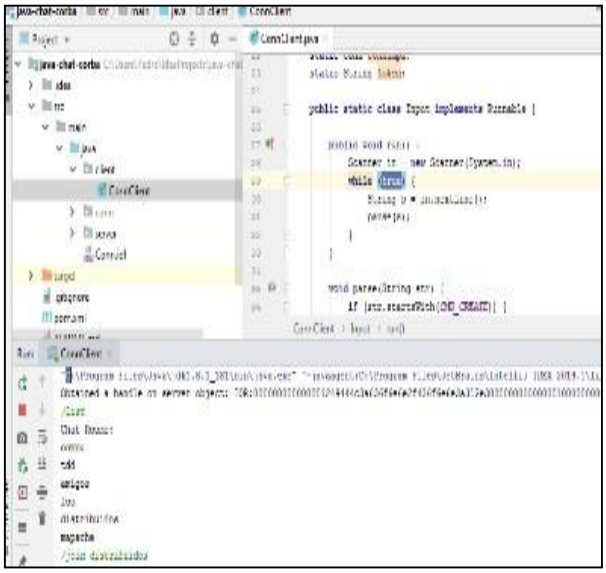

Figura 6. Interfaz de IntelliJ.

En un primer intento de establecer la comunicación entre varios clientes, desde la consola de InteliJ, en la figura 7 se muestra una pequeña conversación entre los integrantes del grupo. Previamente se han realizado los pasos anteriores, para obtener resultados en consola.

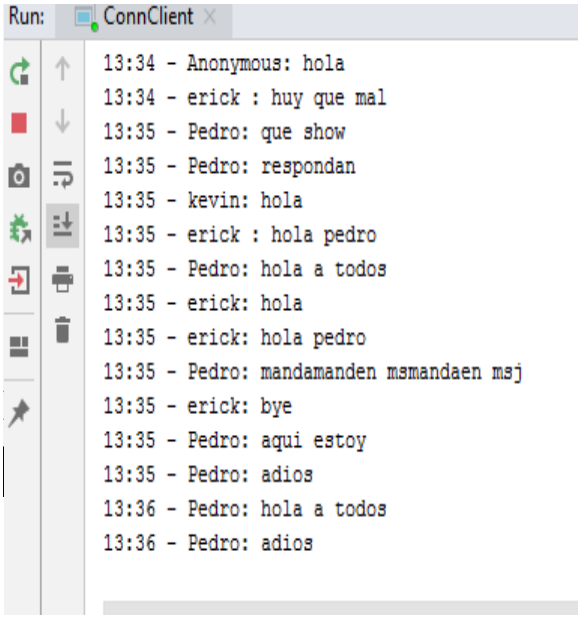

Figura 7. Corrida del programa en IntelliJ

Existen los siguientes comandos especiales de chat:

/create foo. - crea una sala de chat llamada foo

/join foo. - se une a una sala de chat llamada foo

/leave. - sale de la sala de chat actual

/list. - enumera todas las salas de chat

/name foo. - cambia el nombre del cliente a foo

/help. - lista todos los comandos

/quit. - abandona el programa

.

8. RESULTADOS

Ejemplo de la conversación desde consola. Como se mencionó anteriormente se estableció una PC como servidor para evitar cualquier interferencia y se habilitaron 3 clientes que se conectaron a la IP y puerto correspondiente para poder realizar la conversación e intercambio de mensajes que se muestran en la Figura 8.

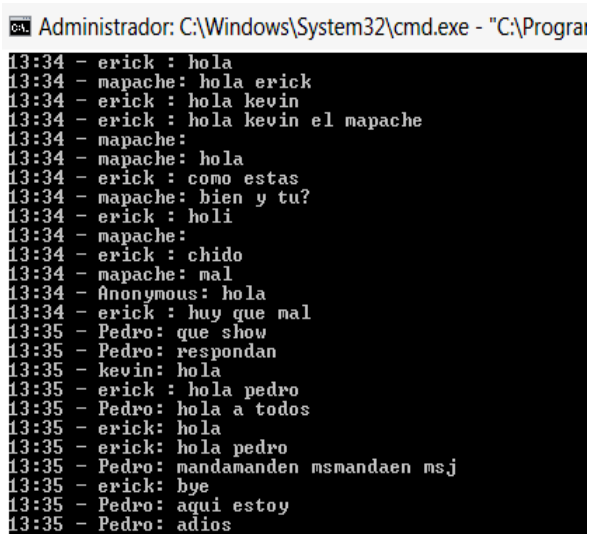

Figura 8. Conversación en cmd.

9. CONCLUSIONES

Después de haber analizado las características propias del lenguaje Java, podemos concluir que, hasta el momento, es un software de programación en el que podemos confiar al diseñar chats, esto es, por su gran versatilidad, facilidad de programación y seguridad.

Los Chat´s en nuestra época están dando un gran giro comercial, ya que no solo se utiliza para intercomunicar a un grupo de gente, sino que también, para establecer una relación Cliente- Servidor, para empresas o instituciones educativas etc.

Actualmente muchas de las grandes empresas ya utilizan los Chat´s para la comunicación con sus clientes, ya que por medio de ellos pueden, cotizar, hacer pedidos de productos, verificar existencias en tiempo real y hasta tener una entrevista "cara a cara" (por medio de videoconferencia). Este tipo de comunicación se puede decir que es más personalizado que la vía telefónica. Por eso es de gran importancia conocer los conceptos básicos y los diferentes avances a lo largo de los años.

El trabajo realizado, se puede ir dándole diferentes cambios, pero estos ya irán relacionados con las necesidades de la empresa o institución que requiera este tipo de proyectos para implementarla en su empresa o negocio.

Referencias

[1] Schaaf, M., & Maurer, F. (2001). Integrating Java and CORBA: A programmer's perspective.IEEE Internet Computing, 5(1), 72-78.

[2] Frantz, D. (2000). CORBA 3 fundamentals and programming(Vol. 2). J. Siegel (Ed.). New York, NY, USA:: John Wiley & Sons.

[3] Felber, P., & Guerraoui, R. (2000). Programming with object groups in CORBA. IEEE Concurrency, 8(1), 48-58.

[4] Schaaf, M., & Maurer, F. (2001). Integrating Java and CORBA: A programmer's perspective. IEEE Internet Computing, 5(1), 72-78.

[5] Jemerov, D. (2008, October). Implementing refactorings in IntelliJ IDEA. In Proceedings of the 2nd Workshop on Refactoring Tools (p. 13). ACM.