

Chat RMI 1.0

1. Enunciado

El objetivo es tener una primera versión que implemente un chat en modo texto (tanto servidor como cliente), con una solución basada en Java RMI. El chat permitirá la comunicación remota entre varios clientes, utilizando un servidor central que recoge los mensajes, y reenvía los mismos a todos los participantes.

La implementación se realizará con Java RMI, con carga dinámica de clases desde un servidor web, tanto en cliente como en servidor. El servidor se implementará siguiendo un modelo *push*, donde el servidor guarda registro de los clientes, y reenvía los mensajes recibidos, uno a uno, a todos los participantes actuales.

Por motivos de simplicidad, los usuarios se registran con un apodo (*nickname*) y se supondrá que nunca se registran dos usuarios con el mismo.

Se manejarán dos tipos básicos de mensaje:

- *texto libre*: texto introducido por el cliente para retransmitir a todos los otros clientes conectados
- *logout*: comando para cerrar la conexión y salir de la aplicación cliente.

Por otro lado, un usuario (cliente) tendrá la posibilidad de bloquear los mensajes de otro usuario incluso si el usuario a bloquear no está todavía conectado. Para realizar este bloqueo se implementará el comando “*ban*”, pudiendo volver a aceptar sus mensajes con el comando “*unban*”. Cuando un usuario haya sido bloqueado, simplemente no se mostrarán sus mensajes. Ejemplo:

- *ban usuariobaneado*
- *unban usuariounbaneado*

Cuando un usuario bloquea a otro, se deberá comunicar este hecho al servidor a través de un mensaje del tipo “*usuarioactual ha baneado a usuariobaneado*”.

La estructura de paquetes y módulos/clases es la siguiente (ver Tabla 1):

Paquete	Módulos / Clases	Nº	Descripción
es.ubu.lsi.client	ChatClient ChatClientDynamic ChatClientImpl ChatClientStarter	1 interfaz (remota) 3 clases	Clases para el cliente.
es.ubu.lsi.common	ChatMessage	1 clase	Clases comunes.
es.ubu.lsi.server	ChatServer ChatServerDynamic ChatServerImpl ChatServerStarter	1 interfaz (remota) 3 clases	Clases para el servidor.

Tabla 1: Resumen de paquetes y módulos

El código fuente de ChatClient.java, ChatClientDynamic.java, ChatMessage.java, ChatServer y ChatServerDynamic están ya disponibles¹.

Es labor de los alumnos implementar los 4 ficheros necesarios para el correcto cierre y ensamblaje del sistema, junto con el resto de productos indicados en el apartado 3 Normas de Entrega.

Para su resolución se dan las siguientes indicaciones. Es decisión del alumno **la inclusión adicional de atributos y/o métodos**.

¹Disponibles en UBUVirtual.

1.1. Paquete *es.ubu.lsi.client*

Comentarios respecto a la interfaz `ChatClient`:

- Define la interfaz remota a implementar por los clientes.

Comentarios respecto a `ChatClientImpl`:

- Implementa la interfaz remota `ChatClient`.

Comentarios respecto a `ChatClientDynamic`:

- Implementa la configuración de un cliente dinámico que carga la clase `ChatClientStarter` para iniciar la ejecución del cliente, cargando sus clases dinámicamente desde un servidor web.

Comentarios respecto a `ChatClientStarter`:

- Recibe como argumentos el *nickname* y el *host* remoto (este último parámetro opcional).
- Inicia el proceso de exportación del cliente remoto y de la resolución del servidor de chat remoto, enlazando ambos objetos (método `checkIn`).
- En un bucle, recibe las entradas de texto del usuario y en función del mismo, invoca a los métodos remotos del servidor.
- Cuando el usuario introduce el texto `logout` finaliza su ejecución.

1.2. Paquete *es.ubu.lsi.common*

Comentarios respecto a la clase `ChatMessage`:

- Mensaje que se envía y recibe en el chat. Encapsula el identificador del cliente que genera el mensaje, el tipo y en algunos casos el texto a mostrar.

1.3. Paquete *es.ubu.lsi.server*

Comentarios respecto a la interfaz `ChatServer`:

- Define la interfaz remota a implementar por el servidor.

Comentarios respecto a la clase `ChatServerImpl`:

- Implementa la interfaz remota `ChatServer` (el método `shutdown` no se implementa en esta práctica dejándose vacío su cuerpo de método).

Comentarios respecto a la clase `ChatServerDynamic`:

- Implementa la configuración de un servidor dinámico que carga la clase `ChatServerStarter` para iniciar la ejecución del servidor, cargando sus clases dinámicamente desde un servidor web.

Comentarios respecto a la clase `ChatServerStarter`:

- Inicia el proceso de exportación del servidor remoto y su registro en RMI (`rmiRegistry`).

1.4. Estructura de proyectos Maven

Para el desarrollo de la práctica se crearán dos proyectos con Maven:

- El primero con nombre `Practica2ChatRMI` y arquetipo `maven-archetype-quickstart`.
 - Contendrá **todas** las clases desarrolladas en la práctica (ver Tabla 1. Resumen de paquetes y módulos) junto con las clases disponibles en UBUVirtual.
 - El fichero `.jar` resultante de invocar a `mvn package` (con nombre `Practica2ChatRMI-0.0.1-SNAPSHOT.jar`) **sólo** contendrá las clases necesarias para arrancar el cliente y el servidor (`ChatClientDynamic` y `ChatServerDynamic`).
 - El resto de clases deben descargarse desde un servidor HTTP.
- El segundo con nombre `Practica2ChatRMI-Web` y arquetipo `webapp-javaee7`.
 - Sólo cumple la función de alojar las clases restantes (no incluidas en el fichero `.jar` previo) para su descarga por HTTP.
 - La aplicación web se desplegará con el nombre `Practica2RMI-Web` sobre el servidor Glassfish.
 - Las clases a descargar cuelgan de dicho directorio raíz de la aplicación web.
 - Ej: para descargar la clases `ChatClientStarter` se debería utilizar la URL:
`http://localhost:8080/Practica2ChatRMI-Web/es/ubu/lsi/client/ChatClientStarter.class`

El fichero `pom.xml` del primer proyecto (`Practica2ChatRMI`) **debe realizar la copia de los `.class`** al correspondiente proyecto web (`Practica2ChatRMI-Web`) y el empaquetado de las clases².

Los ficheros `.bat` de arranque de cliente, servidor y registro se proporcionan en la plataforma UBUVirtual, y deben incluirse en el directorio raíz del proyecto `Practica2ChatRMI`.

2. Normas de Entrega

Fecha límite de entrega:

- **26 de abril del 2023 hasta las 21:30 horas**

Formato de entrega:

Repositorio Github

- Aunque la práctica se haga por parejas, se enviará por parte de cada uno de los dos integrantes.

Para el desarrollo de la práctica se entregarán los dos proyectos con Maven (después de realizar un `mvn clean` en ambos):

- Proyecto `Practica2ChatRMI`.
- Proyecto `Practica2ChatRMI-Web`.

Dentro del proyecto `Practica2ChatRMI` se añadirá además un fichero `build.xml` para su ejecución con ANT que permita la generación de la documentación *javadoc* de dicho proyecto en la carpeta `doc` (target con `name="javadoc"` utilizando el task `javadoc`).

Corrección, defensa de la práctica y comentarios adicionales.

- La práctica se realizará por grupos de **dos personas máximo o bien individualmente**, pudiéndose solicitar **una defensa** de las mismas en horario de prácticas **La no realización de la defensa si se solicita, supondría una calificación de cero.**
- Para aclarar cualquier duda utilizar el foro habilitado. En caso de no aclararse las dudas en el foro, dirigirse al profesor en horario de tutorías.

²Revisar los ejemplos vistos en clases.

- No se deben modificar los ficheros fuente proporcionados. En caso de ser necesario, por errores en el diseño / implementación de los mismos, se notificará para que sea el responsable de la asignatura quien publique en UBUVirtual la corrección y/o modificación. Se modificará el número de versión del fichero en correspondencia con la fecha de modificación y se publicará un listado de erratas.

Criterios de valoración.

- La práctica es obligatoria, entendiendo que su no presentación en la fecha marcada, supone una calificación de cero sobre el total de **1 punto** (sobre 10 del total de la asignatura) que se valora la práctica.
- Se valorará negativamente métodos con un número de líneas grande (>50 líneas) sin contar comentarios. Se debe evitar en la medida de lo posible la repetición de código.
- No se admiten ficheros cuya estructura y contenido no se adapte a lo indicado, con una valoración de cero.
- No se corrigen prácticas que no compilen, ni que contengan errores graves en ejecución con una valoración de cero.
- No se admite código no comentado con la especificación para **javadoc** con una valoración de cero.