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 ciertos tipos básicos de mensaje:
- *texto libre*: texto introducido por el cliente para retrasmitir a todos los otros clientes conectados
- *logout:* comando para cerrar la conexión y salir de la aplicación cliente.
- *shutdown*: comando para cerrar la conexión de todos los clientes y terminar.

Por otro lado, un usuario (cliente) tendrá la posibilidad de eliminar a otro usuario del Chat. Para eliminar un usuario se implementará el comando "*drop*". Ejemplo:

- *drop usuarioaeliminar*

El servidor eliminará el cliente y responderá al usuario que ha solicitado la eliminación con uno de los siguientes mensajes:

- "El usuario *usuarioaeliminar* ha sido desconectado": cuando la eliminación del usuario haya tenido éxito.
- "El usuario *usuarioaeliminar* no existe": cuando se haya solicitado la eliminación de un usuario inexistente.

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`	`ChatMesage`	`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.java` y `ChatServerDynamic.java` están ya disponibles[1].

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***.

[1]Disponibles en UBUVirtual.

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

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`.

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 clase ChatClientStarter se debería utilizar la URL:

`http://localhost:8080/Practica2ChatRMI-Web/es/ubu/lsi/client/ChatClient.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 clases[2].

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:

- **28 de abril de 2023 (viernes) hasta las 23:30 horas**

Formato de entrega:

- Se enviará un fichero `.zip` a través de la plataforma **UBUVirtual** completando la tarea **Entrega de Práctica Obligatoria 2 - EPO2.**
- El fichero comprimido seguirá el siguiente formato de nombre:

  `NombrePrimerApelido-NombrePrimerApellido.zip`

  Ej: si los alumnos son Pio López y Blas Pérez, su fichero se llamará:

  `PioLopez-BlasPerez.zip`
- **Aunque la práctica se haga por parejas, se enviará por parte de cada uno de los dos integrantes.**
- **NO se admiten envíos posteriores a la fecha y hora límite, ni a través de otro medio que no sea la entrega de la tarea en UBUVirtual. Si no se respetan las normas de envío la calificación es cero.**

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

- Proyecto `Practica2ChatRMI.`

[2]Revisar los ejemplos vistos en clases.

- 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.
- No se deben modificar los ficheros fuentes proporcionados, si bien, sí que se podrán añadir ciertos atributos o métodos si se considera necesario.

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 (>30 líneas). Se debe evitar en la medida de lo posible la repetición de código.
- Se valorará positivamente la limpieza en el código y la inclusión de comentarios aclaratorios.
- No se admiten ficheros cuya estructura y contenido no se adapte a lo indicado, con una valoración de cero.
- Se valorará positivamente el aspecto más o menos "profesional" que se dé a la aplicación.
- 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.