UCAM

Enunciado práctica 1

Desarrollo de Aplicaciones Distribuidas I

Miguel Ángel Guillén Navarro

Grado en Ingeniería Informática

Índice de contenidos

1. Introducción

El objetivo de la práctica es desarrollar los conceptos teóricos del desarrollo de aplicaciones distribuidas implementado, mediante sockets, algunos de los componentes básicos de un ERP.

Elementos de la práctica

El equipo de directivo de la empresa para la que trabajamos ha contactado con el gerente de una empresa de desarrollo software y ha llegado a un acuerdo para que seamos nosotros los que implementemos la aplicación que comercializa. Entre otros muchos servicios la empresa oferta la posibilidad de gestionar clientes, productos, personal, facturas o leads[1].

Aunque actualmente la empresa sólo tiene una sede en un futuro sí tiene pensado ampliar sus servicios a través de una red de clientes. Por eso se estima que lo mejor es tener un servidor donde se almacene toda la información y que cada sede se conecte a éste para enviar y recibir los datos.

El proceso que hará las veces de cliente se conectará al servidor para enviar y recibir los comandos correspondientes a la gestión de creación, eliminación y recuperación de los datos.

[1] Un lead representa un potencial cliente

2. Protocolo a implementar

El protocolo de comunicación entre cliente y servidor establece la arquitectura del sistema así como los comandos necesarios para su gestión.

También cuenta con un control de acceso (usuario y contraseña). De tal forma que sólo los usuarios registrados pueden acceder a toda la funcionalidad de la aplicación. Para simplificar dicho funcionamiento sólo se utilizará el usuario *admin* con clave *admin*.

En cuanto a la arquitectura hay que destacar que se trata de un protocolo de tipo petición-respuesta y que tiene dos canales de comunicación, uno para comandos y otro para datos. El primer canal trabaja sobre el puerto 2022 mientras que el canal de envío de datos lo hace sobre el 2023 y sucesivos.

El cliente debe estar autenticado para que el servidor pueda ejecutar cualquiera de los comandos de gestión que se le envían.

2.1 Especificación de los comandos a implementar

Especificación	Descripción	Respuestas
<number> USER <name>	Envía el nombre del usuario con el que realizar la conexión.	OK <number> <codrespuesta> Envíe contraseña. FAILED <number> <codrespuesta> Not user.
<number> PASS <pass>	Envía la contraseña del usuario.	OK <number> <codrespuesta> Welcome <name> FAILED <number> <codrespuesta> Prueba de nuevo
<number> EXIT	Cierra la sesión	OK <number> <codrespuesta> Bye
Comandos de gestión de clientes		
<number> ADDCLIENTE	Solicita añadir un cliente. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> UPDATECLIENTE <id>	Solicita actualizar los datos de un cliente identificado con <id>. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.

<number> GETCLIENTE <id>	Solicita los datos de un cliente identificado con <id>. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> REMOVECLIENTE <id>	Solicita eliminar el cliente identificado con <id>	OK <number> <codrespuesta> Cliente eliminado. FAILED <codrespuesta> No autorizado.
<number> LISTCLIENTES	Listado de clientes existentes. Los datos se recibirán en un array por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> COUNTCLIENTES	Solicita el total de clientes que hay en el sistema	OK <number> <codrespuesta> <total> FAILED <number> <codrespuesta> Mensaje error.
Comandos de gestión de facturas		
<number> ADDFACTURA	Solicita añadir una factura. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.

<number> GETFACTURA <id>	Solicita los datos de una factura identificada con <id>. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> REMOVEFACTURA <id>	Solicita eliminar la factura identificada con <id>	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <codrespuesta> No autorizado.
<number> LISTFACTURAS	Listado de facturas existentes. Los datos se recibirán en un array por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> ADDCLIENTE2FACT <idcliente> <idfactura>	Solicita añadir un cliente(<idcliente>) a una factura(<idfactura>)	OK <number> <codrespuesta> Cliente añadido a la factura FAILED <number> <codrespuesta> Mensaje error.
<number> REMOVECLIFROMFACT <idcliente> <idfactura>	Solicita eliminar un cliente (<idvacuna>) de una factura (<idfactura>)	OK <number> <codrespuesta> Cliente eliminado FAILED <number> <codrespuesta> Mensaje error.
<number> ADDPROD2FACT <idproducto> <idfactura>	Solicita añadir un producto(<idproducto>) a una factura (<idfactura>)	OK <number> <codrespuesta> Producto añadido FAILED <number> <codrespuesta> Mensaje error.

<number> REMOVEPRODFROMFACT <idproducto> <idfactura>	Solicita eliminar un producto (<idproducto>) de una factura (<idfactura>).	OK <number> <codrespuesta> Producto eliminado FAILED <number> <codrespuesta> Mensaje error.
	Comandos de gestión de productos	
<number> ADDPRODUCTO	Solicita añadir un producto. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> UPDATEPRODUCTO <id>	Solicita actualizar los datos del producto identificado con <id>. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> GETPRODUCTO <id>	Solicita los datos del producto identificado con <id>. Los datos se enviarán por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> REMOVEPRODUCTO <id>	Solicita eliminar el producto identificado con <id>	OK <number> <codrespuesta> Producto eliminado. FAILED <codrespuesta> Mensaje error.

<number> LISTPRODUCTOS	Listado de productos existentes. Los datos se recibirán en un array por el canal de datos. El servidor responde con la ip y el puerto donde conectarse.	PREOK <number> <codrespuesta> <ip> <puerto> OK <number> <codrespuesta> Transferencia terminada FAILED <number> <codrespuesta> Mensaje error.
<number> COUNTPRODUCTOS	Solicita el total de productos que hay en el sistema	OK <number> <codrespuesta> <total> FAILED <number> <codrespuesta> Mensaje error.

2.2 Detalles sobre el protocolo

Como se ha comentado anteriormente se trata de un protocolo de tipo petición-respuesta por lo que para cada envío del cliente el servidor responderá, al menos, con otro envío.

La estructura general del paquete que envía el cliente es:

<number> <comando> <información adicional>

Mientras que la respuesta del servidor sigue esta otra:

<tipo respuesta> <number> <código respuesta> <información adicional>

¿Qué es <*number*>? Cada envío/mensaje enviado por el cliente hacia el servidor está identificado por un texto alfanumérico (números, letras o números y letras). No se trata de un identificador del comando sino del envío en sí. Por ejemplo, para esta secuencia de envíos:

1 USER nombre

2 PASS clave

3 ADDX

4 ADDX

5 LISTX

Se puede observar que el comando *ADDX* está repetido dos veces, pero el número que lo precede es diferente ya que se trata del número de envío no del identificador del comando.

El número lo genera el cliente. El servidor sólo lo utiliza para identificar el envío y poder enviar la respuesta correcta al envío del cliente.

¿Qué es <*comando*>? Cada una de las acciones que el cliente puede solicitar que ejecute el servidor. En el apartado anterior se han descrito todos ellos.

¿Qué es *<tipo respuesta>*? Cada comando enviado tendrá una respuesta por parte del servidor. El primer elemento de dicha cadena de caracteres que indica el tipo de respuesta enviada. Pueden ser tres valores:

OK: La ejecución ha sido correcta. Ya no hay más respuestas para ese número de envío (<number>).

PREOK: La ejecución ha sido correcta, pero hay más respuestas para ese número de envío. Esto suele suceder con los comandos que implican el uso del canal de datos. En la primera respuesta se envía IP y puerto mientras que en la segunda se envía el resultado de la transmisión.

FAILED: La ejecución ha sido incorrecta. Ya no hay más respuestas para ese número de envío (<number>).

¿Qué es *<código respuesta>*? De cara a detallar mejor el resultado de la ejecución de un comando el servidor envía un código específico. Los números pueden tener varios dígitos pero siempre empiezan por 2 los que implican una ejecución correcta del comando y por 4 los que han fallado. Se deja a criterio del alumno el listado de los mismos.

Envío por el canal de datos

Para el envío de datos no hace falta transformar los valores de los atributos en una cadena. Se podrá enviar el objeto a través del socket directamente. La secuencia de código para escribir un objeto es:

```
Object object = new Object();

ObjectOutputStream oos = new
ObjectOutputStream(socket.getOutputStream());

oos.writeObject(object);

oos.flush();
```

Importante que la clase cuyo objeto se va a enviar implemente la interfaz *Serializable*.

Para leer dicho objeto en el otro extremo del socket se utilizará el método *readObject* de la clase *ObjectInputStream*.

```
ObjectInputStream ios =
        new ObjectInputStream(socket.getInputStream());
object = (Object) ios.readObject();
```

La creación del canal de datos se realiza siempre de la misma manera:

1. El cliente envía un comando que necesita de transmisión de datos.
2. El servidor envía como respuesta (PREOK) a través del canal de comandos la ip y el puerto al que debe conectarse. Para ello tendrá un listado de puertos disponibles (del 2023 en adelante) y cogerá uno para crear el ServerSocket correspondiente.
3. El servidor se pone a la escucha *serverSocket.accept()*.
4. El cliente lanza una petición de conexión, *new Socket(ip,puerto)*, a la ip y el puerto que el servidor le indicó.
5. Se procede al envío y recepción del objeto solicitado.

2.3 Sistema de control de usuarios conectados

De manera opcional se podrá implementar un canal de comunicación entre cliente y servidor para que este último tenga constancia de que el cliente sigue activo.

La idea es que cada 30 segundos el servidor envíe un mensaje de KEEPALIVE a través de dicho canal de comunicación. El cliente debe responder OK y si no lo hace el servidor lo quitará del listado de clientes conectados.

Todos los clientes podrán enviar por el canal de comandos el mensaje CONECTADOS y el servidor responderá con el número de usuarios que hay actualmente con sesión activa.

3. Funcionalidad mínima exigida

3.1 Entrega parcial

1. Están implementados de forma independiente cliente y servidor; sólo se comunican haciendo uso del protocolo propuesto.
2. Se han implementado los comandos USER, PASS, ADDCLIENTE y GETCLIENTE.

En esta entrega no es necesaria interfaz gráfica, las operaciones se realizarán por consola.

3.2 Entrega final enero

1. Están implementados de forma independiente cliente y servidor; sólo se comunican haciendo uso del protocolo propuesto.
2. Se gestionan los comandos para clientes y productos.

3.3 Entrega julio

1. La misma que en la convocatoria anterior y además:
    a. Se han implementado los comandos para la gestión de facturas.

3.4 Convocatoria especial

Dicha convocatoria se llevará a cabo en el curso 22-23.

1. La misma que en la convocatoria anterior y además:
    a. Se añade una gestión del stock de productos.

4. Entregas que realizar

4.1 Entrega parcial

¿Qué? → Un zip nombrado con DNI y apellidos que incluya el proyecto Eclipse y el diagrama de clases implementadas.

¿Cuándo? → Fecha tarea campus.

¿Dónde? → Campus virtual, tarea “Enero Práctica 1 Sockets – Entrega parcial”.

4.2 Entrega final

¿Qué? → Un zip nombrado con el DNI y apellidos que incluya: el proyecto Eclipse (cuyo nombre también será el DNI), el autoinforme y un documento con la memoria.

¿Cómo? → En el campus virtual, siguiendo las especificaciones propuestas en el documento “Formato documentación prácticas” que se encuentra en la común y en el campus virtual.

¿Cuándo? → Fecha tarea campus.

¿Dónde? → Campus virtual, tarea “Enero Práctica 1 Sockets”.

En septiembre será “Julio Práctica 1 Sockets” y en convocatoria especial “Convocatoria Especial Práctica 1 Sockets”.

5. Criterios de evaluación y calificación

A la hora de evaluar la práctica se tendrán en cuenta varios criterios que determinan el grado en la calificación; estos son:

- ✓ La estructura de clases es correcta y cumple con los principios de la Programación Orientada a Objetos.
- ✓ Existe una explicación fundamentada para la estructura de clases diseñada.
- ✓ El código está optimizado.
- ✓ El código está comentado.
- ✓ Existe un correcto control de errores tanto a nivel de excepción en la ejecución como de interfaz de usuario.
- ✓ Se sigue el modelo en dos capas propuesto en esta memoria.

Antes de calificar se comprueba que:

- ✓ La memoria esté bien redactada y estructurada.
- ✓ La funcionalidad mínima haya sido implementada.

La puntuación que se puede obtener en cada entrega es 1 y 10 respectivamente. Siendo la nota final de la asignatura la suma de las dos. Por tanto, la primera entrega sirve como punto extra.

5.1 Autoinforme

En el autoinforme, sólo necesario para la segunda entrega, se especifica la funcionalidad a implementar, así como los puntos que se pueden obtener como máximo. En la tabla anterior se especifican los criterios por los que se evalúa la funcionalidad exigida y siempre teniendo en cuenta el máximo posible a partir del autoinforme.

Por ejemplo, si en el autoinforme se incluye funcionalidad para optar a un 8 como máximo los porcentajes se aplican sobre esa nota como tope. Por tanto, si sacas un 10 en cada concepto (Memoria, Estructura de clases, etc.) tendrás un 8.

Además de los puntos básicos que sirven para establecer la máxima puntuación existen una serie de puntos extra. Estos se sumarán al resultado de aplicar el porcentaje. La suma nunca superará el máximo establecido. En el ejemplo anterior, aunque se hubiera implementado una funcionalidad extra no se pasaría del 8. En cambio, si al aplicar los porcentajes obtienes un 6 pero has hecho funcionalidad extra para sumar un punto la nota final será un 7.