Universidad Tecnológica Nacional
Facultad Regional Santa Fe

TRABAJO PRÁCTICO
SISTEMAS OPERATIVOS DISTRIBUIDOS

Docente: Pablo Pessolani
Alumno: Agustín Tibaldo

SOCKET TCP

cliente.java

```
import java.net.*;
import java.io.*;

public class cliente {

	static final String ALUMNO = "A";
	static final String PROFESOR = "P";

	public static void main(String[] args) throws IOException {

		int puerto = 0, libreta = 0, nota = 0;
		String usuario = null;

		//////////////////////////
		// Parsing parameters
		//////////////////////////
		for (int i = 0; i < args.length; i++) {
			switch (args[i]) {
			case "-p":
				puerto = Integer.parseInt(args[i+1]);
				break;
			case "-A":
				usuario = ALUMNO;
				break;
			case "-P":
				usuario = PROFESOR;
				break;
			}

			//opt1: LU
			if (i == 3)
				libreta = Integer.parseInt(args[i]);

			//opt2: Nota
			if (i == 4)
				nota = Integer.parseInt(args[i]);
		}
		//////////////////////////
```

```
            Socket socketCliente = null;
            BufferedReader entrada = null;
            PrintWriter salida = null;

            // Create socket
            try {
                  socketCliente = new Socket("localhost",puerto);
                  entrada = new BufferedReader(new
InputStreamReader(socketCliente.getInputStream()));
                  salida = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socketCliente.getOutputStream())),true);
            } catch (IOException e) {
                  System.err.println("No puede establecer canales de E/S
para la conexión");
                  System.exit(-1);
            }

            BufferedReader stdIn =
                        new BufferedReader(new
InputStreamReader(System.in));

            String linea;

            try {

                  // USUARIO
                  linea = usuario;
                  salida.println(linea);

                  //LU
                  linea = Integer.toString(libreta);
                  salida.println(linea);

                  //Nota
                  linea = Integer.toString(nota);
                  salida.println(linea);

                  // RESPUESTA DEL SERVIDOR
                  linea = entrada.readLine();
                  if (usuario.equals(ALUMNO)) {
                        /*ALUMNO*/
                        if (linea.equals("null")) {
                              System.out.printf("La libreta %s no
corresponde a ningún alumno.\n",libreta);
                        } else {
```

```
                              System.out.printf("La nota del alumno con
LU %s es %s\n", libreta, linea);
                        }
                  } else {
                        /*PROFESOR*/
                        if (linea.equals("-1")) {
                              System.out.printf("Se produjo un error al
intentar guardar la nota del alumno LU %s",libreta);
                        } else {
                              System.out.printf("La nota del alumno LU %s
se ha guardado con éxito!",libreta);
                        }
                  }

            } catch (IOException e) {
                  System.out.println("IOException: "+ e.getMessage());
            }

            // Free resources
            salida.close();
            entrada.close();
            socketCliente.close();
      }

}
```

servidor.java

```
import java.io.*;
import java.net.*;
import java.util.HashMap;
import java.util.Map;

public class Server {

	static final String ALUMNO = "A";
	static final String PROFESOR = "P";

	public static void main(String[] args)  throws IOException {
		int puerto = 0;
		String usuario = null, libreta = null, nota = null;

		//////////////////////////
		// Parsing parameters
		//////////////////////////
		for (int i = 0; i < args.length; i++) {
			switch (args[i]) {
			case "-p":
				puerto = Integer.parseInt(args[i+1]);
				break;
			}
		}
		//////////////////////////
		ServerSocket socketServidor = null;
		try {
			socketServidor = new ServerSocket(puerto);

		} catch (IOException e) {
			System.out.println("No puede escuchar en el puerto: "+
puerto);
			System.exit(-1);
		}

		Socket socketCliente = null;
		BufferedReader entrada = null;
		PrintWriter salida = null;

		BufferedReader stdIn =
				new BufferedReader(new
InputStreamReader(System.in));
```

```
            Map<String, String> map = new HashMap<String, String>();
            map.put("100", "9");
            map.put("200", "4");
            map.put("300", "7");

            String linea = null;

            System.out.println("Escuchando: "+ puerto);
            try {
                  while (true) {
                        // Se bloquea hasta que recibe peticion del
cliente
                        // abriendo un socket para el cliente
                        socketCliente = socketServidor.accept();
                        System.out.println("Conexion aceptada: "+
socketCliente);

                        //Establece canal de entrada
                        entrada = new BufferedReader(new
InputStreamReader(socketCliente.getInputStream()));

                        // Establece canal de salida
                        salida = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socketCliente.getOutputStream())),true);

                        usuario = entrada.readLine();
                        System.out.println("Usuario: "+ usuario);
                        libreta = entrada.readLine();
                        System.out.println("Libreta: "+ libreta);
                        nota = entrada.readLine();
                        System.out.println("Nota: "+ nota);

                        if (usuario.equals(ALUMNO)) {
                              /*ALUMNO*/
                              salida.println(map.get(libreta));
                        } else {
                              /*PROFESOR*/
                              try {
                                    map.put(libreta, nota);
                                    salida.println("0");
                              } catch (Exception e) {
                                    salida.println("-1");
                              }
                        }
```

```
                        linea = stdIn.readLine();
                        if (linea.equals("exit")) {
                              break;
                        }
                  }

            } catch (IOException e) {
                  System.out.println("IOException: " + e.getMessage());

            }

            salida.close();
          entrada.close();
          socketCliente.close();
          socketServidor.close();
      }
}
```

REMOTE PROCEDURE CALL (RPC)

cliente.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import org.apache.xmlrpc.client.*;
import org.apache.xmlrpc.common.*;
import org.apache.xmlrpc.*;
import org.apache.xmlrpc.client.XmlRpcClient;

public class cliente {

	static final String ALUMNO = "A";
	static final String PROFESOR = "P";

	public static void main(String[] args) throws IOException {

		int puerto = 0, libreta = 0, nota = 0;
		String usuario = null;

		//////////////////////////
		// Parsing parameters
		//////////////////////////
		for (int i = 0; i < args.length; i++) {
			switch (args[i]) {
			case "-p":
				puerto = Integer.parseInt(args[i+1]);
				break;
			case "-A":
				usuario = ALUMNO;
				break;
			case "-P":
				usuario = PROFESOR;
				break;
			}

			//opt1: LU
			if (i == 3)
```

```
                    libreta = Integer.parseInt(args[i]);

                //opt2: Nota
                if (i == 4)
                    nota = Integer.parseInt(args[i]);

            }
            ////////////////////////

            try {
               XmlRpcClientConfigImpl config = new
XmlRpcClientConfigImpl();
               config.setServerURL(new
URL("http://127.0.0.1/minimatero"));

               // create the client and configure it with
instantiated configuration
               XmlRpcClient server = new XmlRpcClient();
               server.setConfig(config);

                // Cargo parámetros para pasar a server
                Vector params = new Vector();
                params.addElement(libreta);
                params.addElement(nota);
                params.addElement(usuario);

                  if (usuario.equals(ALUMNO)) {
                      /*ALUMNO*/
                     Object result = server.execute("matero.get",
params);
                      if (result.equals("null")) {
                           System.out.printf("La libreta %s no
corresponde a ningún alumno.\n",libreta);
                      } else {
                           System.out.printf("La nota del alumno
con LU %s es %s\n", libreta, result);
                      }
                  } else {
                      /*PROFESOR*/
                      Object result = server.execute("matero.add",
params);
```

```
                    if (result.equals("-1")) {
                         System.out.printf("Se produjo un error
al intentar guardar la nota del alumno LU %s",libreta);
                    } else {
                         System.out.printf("La nota del alumno
LU %s se ha guardado con éxito!",libreta);
                    }
                }

           } catch (Exception e) {
                System.out.println("IOException: "+
e.getMessage());
           }

     }

}
```

servidor.java

```
import java.io.*;
import java.net.*;
import java.util.HashMap;
import java.util.Map;
import org.apache.xmlrpc.*;
import org.apache.xmlrpc.webserver.WebServer;

public class server {

	static final String ALUMNO = "A";
	static final String PROFESOR = "P";

	static Map<String, String> map = new HashMap<String,
String>();

	// GET: devuelvo nota de libreta
	public String get(String LU){
		return map.get(LU);
	}

	// ADD: inserto nota asociada a LU
	public String add(String LU, String nota){
		try {
			map.put(LU, nota);
			return "0";
		} catch (Exception e) {
			return "1";
		}
	}

	public static void main(String[] args)  throws IOException {
		int puerto = 0;
		String usuario = null, libreta = null, nota = null;

		//////////////////////////
		// Parsing parameters
		//////////////////////////
		for (int i = 0; i < args.length; i++) {
			switch (args[i]) {
```

```
                case "-p":
                      puerto = Integer.parseInt(args[i+1]);
                      break;
                }
          }
          ////////////////////////
          map.put("100", "9");
          map.put("200", "4");
          map.put("300", "7");

          try {
             System.out.println("Intentando conectar al servidor
RPC en puerto "+ puerto);

             WebServer server = new WebServer(80);
             server.addHandler("matero", new server());
             server.start();

             System.out.println("Conexión exitosa!");
             System.out.println("Escuchando peticiones...");

          } catch (IOException e) {
                System.out.println("IOException: " +
e.getMessage());
          }

     }
}
```

SPREAD TOOLKIT

cliente.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import spread.*;

public class cliente implements BasicMessageListener {

	static final String ALUMNO = "A";
	static final String PROFESOR = "P";
	static final String GRUPO_CLIENTE = "GRUPO_CLIENTE";
	static final String GRUPO_SERVIDOR = "GRUPO_SERVIDOR";
	private static SpreadConnection conexionSpread;
	private static String userName;
	private static int libreta = 0;
	private static String usuario = null;

	////////////////////////////////////
    /// CONEXION SPREAD
    ////////////////////////////////////
	private static void ConexionSpread(String nombreCliente,
String ip, int puerto) {

		try
		{
			// Conecto
			conexionSpread = new SpreadConnection();
			conexionSpread.connect(InetAddress.getByName(ip),
puerto, nombreCliente, false, true);

			// Unión a grupo
			SpreadGroup spreadGroup = new SpreadGroup();
			spreadGroup.join(conexionSpread, GRUPO_CLIENTE);
			System.out.println("Uniendo a " + spreadGroup +
".");
			System.out.println("Se ha establecido la conexión
" + nombreCliente + " con " + ip + ":" + puerto);
```

```
            }
            catch(SpreadException e)
            {
                  System.err.println("Ha ocurrido un error al
conectar a Daemon");
                  e.printStackTrace();
                  System.exit(1);
            }
            catch(UnknownHostException e)
            {
                  System.err.println("El Daemon de Spread no fue
encontrado." + ip);
                  System.exit(1);
            }

      }

      ////////////////////////////////////
    /// CERRAR CONEXION SPREAD
    ////////////////////////////////////
      private static void CerrarConexionSpread() {

            try {
                  conexionSpread.disconnect();
            } catch (SpreadException e) {
                  e.printStackTrace();
            }

      }

      ////////////////////////////////////
    /// Override MESSAGERECEIVED
    ////////////////////////////////////
      @Override
      public void messageReceived(SpreadMessage msg) {
            try{
                  if (msg.isRegular()){
                        byte data[] = msg.getData();
                        String linea = new String(data);
```

```
                        // RESPUESTA DEL SERVIDOR
                        if (usuario.equals(ALUMNO)) {
                              /*ALUMNO*/
                              if (linea.equals("null")) {
                                    System.out.printf("La libreta %s
no corresponde a ningún alumno.\n",libreta);
                              } else {
                                    System.out.printf("La nota del
alumno con LU %s es %s\n", libreta, linea);
                              }
                        } else {
                              /*PROFESOR*/
                              if (linea.equals("-1")) {
                                    System.out.printf("Se produjo un
error al intentar guardar la nota del alumno LU %s",libreta);
                              } else {
                                    System.out.printf("La nota del
alumno LU %s se ha guardado con éxito!",libreta);
                              }
                        }
                  }
            }
            catch(Exception e)
            {
                  e.printStackTrace();
                  System.exit(1);
            }


      }

      ////////////////////////////////////
     /// MAIN
     ////////////////////////////////////
      public static void main(String[] args) throws IOException {

            int puerto = 0, nota = 0;
            String nombreCliente = "mm_cliente";
            String ip = "127.0.0.1";

            ////////////////////////
```

```
        // Parsing parameters
        ////////////////////////
        for (int i = 0; i < args.length; i++) {
              switch (args[i]) {
              case "-p":
                    puerto = Integer.parseInt(args[i+1]);
                    break;
              case "-A":
                    usuario = ALUMNO;
                    break;
              case "-P":
                    usuario = PROFESOR;
                    break;
              }

              //opt1: LU
              if (i == 3)
                    libreta = Integer.parseInt(args[i]);

              //opt2: Nota
              if (i == 4)
                    nota = Integer.parseInt(args[i]);

        }
        ////////////////////////
        String linea;

        // Conecto con SPREAD
        ConexionSpread(nombreCliente, ip, puerto);

      /////////
        // Envío parámetro Usuario
        linea = usuario;
      SpreadMessage msgUsuario = new SpreadMessage();
        msgUsuario.setSafe();
        msgUsuario.addGroup(GRUPO_SERVIDOR);
        msgUsuario.setData(linea.getBytes());
        try {
              conexionSpread.multicast(msgUsuario);
        } catch (SpreadException e) {
              e.printStackTrace();
```

```
            }

            /////////
           // Envío parámetro Libreta
            linea = Integer.toString(libreta);;
         SpreadMessage msgLU = new SpreadMessage();
         msgLU.setSafe();
         msgLU.addGroup(GRUPO_SERVIDOR);
         msgLU.setData(linea.getBytes());
            try {
                  conexionSpread.multicast(msgLU);
            } catch (SpreadException e) {
                  e.printStackTrace();
            }

            /////////
           // Envío parámetro Nota
            linea = Integer.toString(libreta);;
         SpreadMessage msgNota = new SpreadMessage();
         msgNota.setSafe();
         msgNota.addGroup(GRUPO_SERVIDOR);
         msgNota.setData(linea.getBytes());
            try {
                  conexionSpread.multicast(msgNota);
            } catch (SpreadException e) {
                  e.printStackTrace();
            }

            // Recibo mensaje de servidor
            RecepcionMensajes recepcion = new
RecepcionMensajes(usuario,libreta);
            recepcion.setConnection(conexionSpread);

            // Cierro conexión SPREAD
            CerrarConexionSpread();
      }

}
```

RecepcionMensajes.java

```
import java.net.InetAddress;
import java.net.UnknownHostException;

import spread.BasicMessageListener;
import spread.SpreadConnection;
import spread.SpreadException;
import spread.SpreadGroup;
import spread.SpreadMessage;

public class RecepcionMensajes implements Runnable,
BasicMessageListener{

	private static SpreadConnection conexionSpread;
	static final String ALUMNO = "A";
	static final String PROFESOR = "P";
	private static int libreta = 0;
	private static String usuario = null;

	//Constructor
	public RecepcionMensajes(String usr, int LU) {
		this.usuario = usr;
		this.libreta = LU;
	}

	public void setConnection(SpreadConnection conn){
		conexionSpread = conn;
	}

	@Override
	public void messageReceived(SpreadMessage msg) {
		try{
			if (msg.isRegular()){
				byte data[] = msg.getData();
				String linea = new String(data);

				// RESPUESTA DEL SERVIDOR
				if (usuario.equals(ALUMNO)) {
					/*ALUMNO*/
					if (linea.equals("null")) {
```

```
                                    System.out.printf("La libreta %s
no corresponde a ningún alumno.\n",libreta);
                              } else {
                                    System.out.printf("La nota del
alumno con LU %s es %s\n", libreta, linea);
                              }
                        } else {
                              /*PROFESOR*/
                              if (linea.equals("-1")) {
                                    System.out.printf("Se produjo un
error al intentar guardar la nota del alumno LU %s",libreta);
                              } else {
                                    System.out.printf("La nota del
alumno LU %s se ha guardado con éxito!",libreta);
                              }
                        }
                  }
            }
            catch(Exception e)
            {
                  e.printStackTrace();
                  System.exit(1);
            }
      }

      @Override
      public void run() {
         SpreadMessage msgreceived = new SpreadMessage();

         boolean esperando = true;

         while(esperando){
            try{
                  msgreceived = conexionSpread.receive();
                  messageReceived(msgreceived);
            }
            catch(Exception e)
            {
                  e.printStackTrace();
                  System.exit(1);
            }
```

```
        }
      }

}
```

server.java

```
import java.io.*;
import java.net.*;
import java.util.HashMap;
import java.util.Map;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;
import spread.*;



public class server implements BasicMessageListener {

      static final String ALUMNO = "A";
      static final String PROFESOR = "P";
      static final String GRUPO_CLIENTE = "GRUPO_CLIENTE";
      static final String GRUPO_SERVIDOR = "GRUPO_SERVIDOR";
      private boolean esPrimario;
      private int backupActual;
      private static SpreadConnection conexionSpread;
      private static String userName;
      private SpreadGroup spreadGroup;
      private static int libreta = 0;
      private static int nota = 0;
      private static String usuario = null;
      private static Map<String, String> map = new HashMap<String,
String>();


      ////////////////////////////////
```

```
    /// CONEXION SPREAD
    ////////////////////////////////////
     private static void ConexionSpread(String nombreservidor,
String ip, int puerto) {

          try
          {
               // Conecto
               conexionSpread = new SpreadConnection();
               conexionSpread.connect(InetAddress.getByName(ip),
puerto, nombreservidor, false, true);

               // Unión a grupo
               SpreadGroup spreadGroup = new SpreadGroup();
               spreadGroup.join(conexionSpread, GRUPO_SERVIDOR);
               System.out.println("Uniendo a " + spreadGroup +
".");
               System.out.println("Se ha establecido la conexión
" + nombreservidor + " con " + ip + ":" + puerto);

          }
          catch(SpreadException e)
          {
               System.err.println("Ha ocurrido un error al
conectar a Daemon");
               e.printStackTrace();
               System.exit(1);
          }
          catch(UnknownHostException e)
          {
               System.err.println("El Daemon de Spread no fue
encontrado." + ip);
               System.exit(1);
          }

     }

     ////////////////////////////////////
    /// Override MESSAGERECEIVED
    ////////////////////////////////////
     @Override
```

```
      public void messageReceived(SpreadMessage message) {
            resolveMessage(message);
      }


      ////////////////////////////////////
     /// HANDLE MEMBERSHIP INFO
     ////////////////////////////////////
      private void handleMembershipInfo(MembershipInfo info)
      {
            if(info.isRegularMembership()) {
                if(info.isCausedByDisconnect()) {
                      turnPrimary(true);
                 }
            }
      }

      ////////////////////////////////////
     /// RESOLVE MESSAGE
     ////////////////////////////////////
      private void resolveMessage(SpreadMessage msg)
      {
            try
            {
                 if(msg.isRegular())
                 {

                      byte data[] = msg.getData();
                      String mensajeCliente = new String(data);
                      String respuesta = new String();

                      char letra = mensajeCliente.charAt(0);
                      if (letra == 'U') {
                            usuario = mensajeCliente;
                      }
                      if (letra == 'L') {
                            libreta =
Integer.parseInt(mensajeCliente);
                      }
                      if (letra == 'N') {
                            nota =
```

```
Integer.parseInt(mensajeCliente);
                        }


                        if (usuario.equals(ALUMNO)) {
                              /*ALUMNO*/
                              respuesta = map.get(libreta);
                        } else {
                              /*PROFESOR*/
                              try {

map.put(Integer.toString(libreta), Integer.toString(nota));
                                    respuesta = "0";
                              } catch (Exception e) {
                                    respuesta = "-1";
                              }
                        }

                        if (this.esPrimario) {

                              sendMessage(respuesta, GRUPO_CLIENTE);
                        }

                  }
                  else if (msg.isMembership())
                  {
                        System.out.println("\nRecepción de mensaje
[DE GRUPO]: ");
                        MembershipInfo info =
msg.getMembershipInfo();
                        handleMembershipInfo(info);

                  }
            }
            catch(Exception e)
            {
                  e.printStackTrace();
                  System.exit(1);
            }
      }
```

```
    ////////////////////////////////////
    /// SEND MESSAGE
    ////////////////////////////////////
     private void sendMessage(String info, String groupName) {

          SpreadMessage msg = new SpreadMessage();
          msg.setSafe();
          msg.addGroup(groupName);

          msg.setData(info.getBytes());
          try {
               conexionSpread.multicast(msg);
          } catch (SpreadException e) {
               e.printStackTrace();
          }

     }

     ////////////////////////////////////
    /// CERRAR CONEXION SPREAD
    ////////////////////////////////////
     private static void CerrarConexionSpread() {

          try {
               conexionSpread.disconnect();
          } catch (SpreadException e) {
               e.printStackTrace();
          }

     }

     ////////////////////////////////////
    /// CONSTRUCTOR
    ////////////////////////////////////
     public server(String nombreServidor, String ip, int puerto)
     {
          this.turnPrimary(false);
          backupActual = -1;

          ConexionSpread(nombreServidor, ip, puerto);
     }
```

```java
    ////////////////////////////////////
    /// TURN PRIMARY
    ////////////////////////////////////
    public void turnPrimary(boolean b) {
        this.esPrimario = b;
    }


    ////////////////////////////////////
    /// MAIN
    ////////////////////////////////////
    public static void main(String[] args) {
        int puerto = 0;
        String usuario = null, libreta = null, nota = null;

        //////////////////////////
        // Parsing parameters
        //////////////////////////
        for (int i = 0; i < args.length; i++) {
            switch (args[i]) {
            case "-p":
                puerto = Integer.parseInt(args[i+1]);
                break;
            }
        }

        //////////////////////////
        map.put("100", "9");
        map.put("200", "4");
        map.put("300", "7");

        String nombreServidor = "mm_servidor";
        String ip = "127.0.0.1";
        new server(nombreServidor, ip, puerto);

        // Cierro conexión SPREAD
        CerrarConexionSpread();
    }
```

```
}
```