PROGRAMACIÓN de SERVICIOS y PROCESOS – 2º DAM

PRUEBA 2ª EVALUACIÓN curso 2019/2020

Nombre y Apellidos:	Puntuación:

1. (Máx: 2ptos.) Indica si son Verdaderas ('V') o Falsas ('F') las siguientes afirmaciones (0,1pto. por cada acierto; -0,1ptos. por cada 2 errores):

	RESP	PREGUNTA
1	V	El protocolo IMAP es más complejo que POP ya que permite visualizar los mensajes de manera remota y no descargando los mensajes como lo hace POP.
2	V	El protocolo HTTP define una serie predefinida de métodos de petición (algunas veces referido como "verbos") que indican la acción a efectuar sobre el recurso identificado, tales como PUT, GET, POST…
3	F	La versión para Linux/Unix, OpenLDAP, es incompatible con su implementación análoga de plataformas Windows, Active Directory.
4	F	En el modelo OSI de interconexión de sistemas abiertos sólo las capas 1 y 2 añaden alguna cabecera a la Unidad de Datos de Protocolo (PDU).
5	V	OAuth permite a un usuario utilizar su información de un sitio (proveedor de servicio) con otro sitio (llamado consumidor) sin compartir toda su identidad.
6	F	SMTP es el protocolo más adecuado para la transferencia de ficheros de más de 50MB.
7	V	Un socket es un descriptor de un punto final de comunicación, es decir, representa un extremo de una comunicación bidireccional a través de una dirección IP y un puerto.
8	F	POP3 facilita el acceso y la disponibilidad total a los correos electrónicos desde cualquier dispositivo ya que los mensajes no son eliminados del servidor.
9	V	El protocolo ECHO fue propuesto originalmente para permitir la comprobación de la red y la medición del tiempo de ida y vuelta en las redes IP.
10	V	El protocolo NTP sirve principalmente para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable.
11	V	LDAP ofrece un formato URL para definir búsquedas de clientes en una infraestructura y obtener los valores de sus atributos publicados.
12	F	En una conexión con un servidor FTP anónimo no es necesario indicar ninguna credencial durante el proceso de autenticación.
13	V	La función más importante del protocolo DNS es "traducir" nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, con el propósito de poder localizar y direccionar estos equipos mundialmente.
14	F	Ninguno de los métodos estudiados para el manejo de conexiones mediante sockets es bloqueante.
15	V	HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies…) para comunicarse.
16	V	El espacio de nombres de dominio tiene una estructura arborescente donde los nodos del árbol son etiquetas. Un nombre de dominio completo de un objeto DNS consiste en la

		concatenación de todas las etiquetas de un camino.
17	F	Los sockets UDP Multicast son igual de fiables que los sockets TCP.
18	V	El tipo de acceso FTP denominado invitado (guest) consiste en permitir que cada usuario se conecte a la máquina mediante sus credenciales, pero evitando que tenga acceso a partes del sistema de archivos que no necesita para realizar su trabajo.
19	V	NTP utiliza el algoritmo de Marzullo con la escala de tiempo Tiempo Universal Coordinado, (UTC), para sincronizar sistemas, e incluye soporte para segundos intercalares.
20	V	El protocolo LDAP accede a directorios organizados en forma de árboles de entradas, las cuales tienen identificadores únicos y un conjunto de atributos definidos en un esquema.

2. (Máx: 4ptos.) Se tiene el proyecto en Netbeans “Ej2_DAM2_Eval2” que contiene los ficheros:

   - `Examen.java` : es un modelo de examen, que consta de un String con el nombre del/de la alumno/a y un Map<String, String> para almacenar cada par <pregunta-repuesta>.
   - `Alumno.java` : es una GUI (ver página siguiente) en la que se mostrarán:
     - en la parte superior todos los mensajes que se envían entre los/las alumnos/as.
     - en la parte central 2 líneas: una para solicitar la respuesta de la pregunta que se marque en el cuadro de texto de esa línea junto con un botón “Solicitar”. Y otra línea en la que se da una respuesta a una pregunta que se marcan en los cuadros de texto correspondientes a esa línea junto con un botón “Dar Respuesta”. Estos apartados funcionan mediante sockets UDP Multicast.
     - en la parte inferior hay una tabla de 10 filas por 2 columnas que corresponde con el examen que enviará cada estudiante al profesor (servidor) por una comunicación TCP y éste responderá a través de un entero con la puntuación obtenida.
   - `Servidor.java` y `HiloServidor.java` : el primero es una GUI (ver página siguiente) en la que se muestra qué alumnos/as han enviado ya su examen y sus puntuaciones. Se encarga de recibir las conexiones TCP por parte de los alumnos/as y lanzar un HiloServidor por cada una cuando son aceptadas. El segundo fichero (HiloServidor.java) queda a la espera de recibir un objeto Examen para proceder a su corrección y luego envía la puntuación obtenida como respuesta al/a la alumno/a correspondiente.

   En este ejercicio se deben implementar ÚNICAMENTE las siguientes funciones:

   `Alumno.establecerConexion() , Alumno.solicitarPregunta(String) , Alumno.responderPregunta(String, String), Alumno.enviarExamen(Examen), Alumno.terminarConexion() y Servidor.recibirConexiones()`

   (el resto de funciones de las clases `Alumno` y `Servidor` NO deben modificarse, al igual que los ficheros `Examen.java` y `HiloServidor.java`). Para todas las funciones que se han de implementar se tiene su documentación (en forma JavaDoc) en los ficheros correspondientes.

Posible solución:

`Alumno.java`

`(...)`

```
    /**
     * Función que crea el socket multicast y se une al grupo
     */
    public static void establecerConexion() {
        try {
            // Se crea el socket multicast
            ms = new MulticastSocket(puerto);
            grupo = InetAddress.getByName("225.0.0.1");// Grupo
            // Nos unimos al grupo
            ms.joinGroup(grupo);
        } catch (IOException ex) {
            System.out.println("IOException: " + ex.getMessage());
```

```
        }
    }

    /**
     * Función que crea un String con el texto: "NOMBRE solicita la pregunta
     * PREG" y lo envía al grupo multicast a través de un DatagramPacket
     *
     * @param pregSolicitada pregunta que solicita el
     */
    private void solicitarPregunta(String pregSolicitada) {
        try {
            String texto = nombre + " solicita la pregunta " + pregSolicitada;
            DatagramPacket paquete = new DatagramPacket(texto.getBytes(), texto.length(),
grupo, puerto);
            ms.send(paquete);
        } catch (IOException ex) {
            System.out.println("IOException: " + ex.getMessage());
        }
    }

    /**
     * Función que crea un String con el texto: "NOMBRE dice: pregunta PREG ->
     * respuesta RESP" y lo envía al grupo multicast a través de un
     * DatagramPacket
     *
     * @param preg pregunta que responde el alumno
     * @param resp respuesta que da el alumno para la pregunta PREG
     */
    private void responderPregunta(String preg, String resp) {
        try {
            String texto = nombre + " dice: pregunta " + preg;
            texto += " -> respuesta " + resp;
            DatagramPacket paquete = new DatagramPacket(texto.getBytes(), texto.length(),
grupo, puerto);
            ms.send(paquete);
        } catch (IOException ex) {
            System.out.println("IOException: " + ex.getMessage());
        }
    }

    /**
     * *
     * Función que realiza la conexión TCP con el servidor, enviando el objeto
     * Examen que se pasa como parámetro y recibiendo la puntuación obtenida a
     * través de un int
     *
     * @param examen que se enviará al servidor para ser puntuado
     * @return entero con la puntuación del examen (recibida desde el servidor)
     * o -1 en caso de error
     */
    private int enviarExamen(Examen examen) {
        String equipoServidor = "127.0.0.1";
        int puertoServidor = 40404;
        Socket socketCliente;
        try {
            socketCliente = new Socket(equipoServidor, puertoServidor);
            ObjectOutputStream oos = new ObjectOutputStream(socketCliente.getOutputStream());
            oos.writeObject(examen);
            InputStream is = socketCliente.getInputStream();
```

```
            DataInputStream dis = new DataInputStream(is);
            int puntuacion = dis.readInt();
            is.close();
            dis.close();
            oos.close();
            socketCliente.close();
            return puntuacion;
        } catch (IOException ex) {
            System.out.println("IOException: " + ex.getMessage());
        }
        return -1;
    }

    /**
     * Función para abandonar el grupo y cerrar el socket multicast
     */
    private void terminarConexion() {
        try {
            ms.leaveGroup(grupo);
        } catch (IOException ex) {
            System.out.println("IOException: " + ex.getMessage());
        } finally {
            ms.close();
            repetir = false;
        }
    }
```

Servidor.java

```
(...)

    /**
     * Función para recibir conexiones desde los alumnos. El servidor se
     * mantiene continuamente a la espera de nuevas conexiones y, cuando acepta
     * alguna, crea un HiloServidor para ella, lo lanza y aumenta el valor de
     * CONEXIONES.
     */
    private static void recibirConexiones() {
        try {
            servidor = new ServerSocket(PUERTO);
            while (true) {
                Socket s;
                s = servidor.accept();// esperando cliente
                HiloServidor hilo = new HiloServidor(s);
                hilo.start();
                CONEXIONES++;
            }
        } catch (IOException e1) {
            System.out.println("IOException: " + e1.getMessage());
        }
    }
```

3. (Máx: 4ptos.) Crear un programa Java que se conecte al servidor FTP en la dirección que marque el profesor con las credenciales aportadas para cada estudiante. Una vez dentro, se tiene un fichero `url.txt` con una dirección de internet y otro fichero `mail.txt` con una dirección de correo electrónico. El programa Java deberá descargar ambos ficheros y además crear un directorio de nombre "examPSP". Con la dirección de internet, se obtendrán sus datos a través de un objeto de la clase `URL` y se volcarán sobre un fichero `datosurl.txt` que se subirá al servidor FTP a la carpeta creada. Por otro lado, con la dirección de correo electrónico se enviará un email mediante el protocolo SMTP con el asunto "Examen Eval2 PSP-Alumno".

Posible solución:

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.URL;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

/**
 *
 * @author luis
 */
public class Ej3_DAM2_Eval2 {

    public static void main(String[] args) {
        try {
            System.out.println("Estableciendo conexión con el Servidor FTP");
            FTPClient clienteFTP = new FTPClient();
            String servidorFTP = "127.0.0.1";
            clienteFTP.connect(servidorFTP);
            System.out.println("Autenticándose ante el Servidor FTP");
            String usuario = "usuario1";
            String contrasena = "usu1";
            boolean login = clienteFTP.login(usuario, contrasena);

            //Descargar los 2 ficheros desde el servidor FTP
            FileOutputStream fos = new FileOutputStream("url.txt");
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            boolean descargaFichUrl = clienteFTP.retrieveFile("url.txt", bos);
            FileOutputStream fos2 = new FileOutputStream("mail.txt");
            BufferedOutputStream bos2 = new BufferedOutputStream(fos2);
```

```
            boolean descargaFichMail = clienteFTP.retrieveFile("mail.txt", bos2);

            //Crear directorio mails en el servidor FTP
            boolean dirCreado = clienteFTP.makeDirectory("examPSP");
            if (dirCreado) {
                clienteFTP.changeWorkingDirectory("examPSP");
            } else {
                System.out.println("Error al crear el directorio examPSP en el servidor
FTP");
            }

            //Obtenemos los ficheros descargados y los procesamos
            File ficheroURL = new File("C:\\xampp\\htdocs\\usuario1\\url.txt");
            File ficheroDatosURL = new File("datosurl.txt");
            FileReader lector = null;
            BufferedReader buffer = null;
            lector = new FileReader(ficheroURL);
            buffer = new BufferedReader(lector);
            String urltxt;
            if ((urltxt = buffer.readLine()) != null) {
                FileWriter escritor = null;
                PrintWriter pw = null;
                escritor = new FileWriter(ficheroDatosURL);
                pw = new PrintWriter(escritor);
                URL url;
                url = new URL(urltxt);

                pw.println("\t getProtocol: " + url.getProtocol());
                pw.println("\t getHost: " + url.getHost());
                pw.println("\t getPath: " + url.getPath());
                pw.println("\t getFile: " + url.getFile());
                pw.println("\t getAuthority: " + url.getAuthority());
                pw.println("\t getUserInfo: " + url.getUserInfo());
                pw.println("\t getRef: " + url.getRef());
                pw.println("\t getQuery: " + url.getQuery());
                pw.println("\t toExternalForm: " + url.toExternalForm());
                pw.println("\t toString: " + url.toString());
                pw.println("\t getDefaultPort: " + url.getDefaultPort());
                pw.println("\t getPort: " + url.getPort());

                pw.flush();
                escritor.flush();
                pw.close();
                escritor.close();

                clienteFTP.setFileType(FTP.ASCII_FILE_TYPE);
                FileInputStream fis = new FileInputStream("datosurl.txt");
                BufferedInputStream bis = new BufferedInputStream(fis);
                boolean subida = clienteFTP.storeFile("datosurl.txt", bis);
                if (subida) {
                    System.out.println("Fichero  datosurl.txt  subido  correctamente  al
servidor FTP.");
                } else {
                    System.out.println("Problema  al  subir  el  fichero  datosurl.txt  al
servidor FTP");
                }
            }
            clienteFTP.logout();
            clienteFTP.disconnect();
```

```
            File ficheroMail = new File("C:\\xampp\\htdocs\\usuario1\\mail.txt");
            FileReader lector2 = null;
            BufferedReader buffer2 = null;
            lector2 = new FileReader(ficheroMail);
            buffer2 = new BufferedReader(lector2);
            String dirmail;
            if ((dirmail = buffer2.readLine()) != null) {
                enviarCorreo("desde@gmail.com",      "password",      "true",      "true",
"smtp.gmail.com", "587", dirmail, "Examen Eval2 PSP", "contenido");
            } else {
                System.out.println("Problema al leer la direccion de email. No se ha
enviado nada.");
            }

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

    }

    public static void enviarCorreo(String usuarioDe,
            String contrasenaUsuarioDe,
            String envioSeguroTLS,
            String autenticacionUsuarioDe,
            String servidorSMTP,
            String puertoServidorSMTP,
            String usuarioA,
            String asunto,
            String cuerpo) {

        System.out.println("Estableciendo las propiedades ...");
        Properties propiedades = new Properties();
        propiedades.put("mail.smtp.starttls.enable", envioSeguroTLS);
        propiedades.put("mail.smtp.auth", autenticacionUsuarioDe);
        propiedades.put("mail.smtp.host", servidorSMTP);
        propiedades.put("mail.smtp.port", puertoServidorSMTP);

        System.out.println("Configurando el autenticador ...");
        Authenticator autenticador = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(usuarioDe, contrasenaUsuarioDe);
            }
        };

        System.out.println("Estableciendo una conexión con el servidor SMTP ...");
        Session sesion = Session.getInstance(propiedades, autenticador);

        try {
            System.out.println("Creando el mensaje ...");
            Message mensaje = new MimeMessage(sesion);
            InternetAddress iaDe = new InternetAddress(usuarioDe);
            mensaje.setFrom(iaDe);
            InternetAddress[] iaA = InternetAddress.parse(usuarioA);
            mensaje.setRecipients(Message.RecipientType.TO, iaA);
            mensaje.setSubject(asunto);
            mensaje.setText(cuerpo);
```

```
            System.out.println("Enviando el mensaje ...");
            Transport.send(mensaje);
            System.out.println("Correo electrónico enviado");

        } catch (MessagingException e) {
            System.out.println("Fallo en el envío del correo electrónico: " +
e.getMessage());
        }
    }

}
```

ENTREGA: se realizará exportando cada proyecto en un .zip distinto y subiéndolos al Moodle del curso. Recordar que el tamaño máximo de los ficheros no puede superar los 10MB.

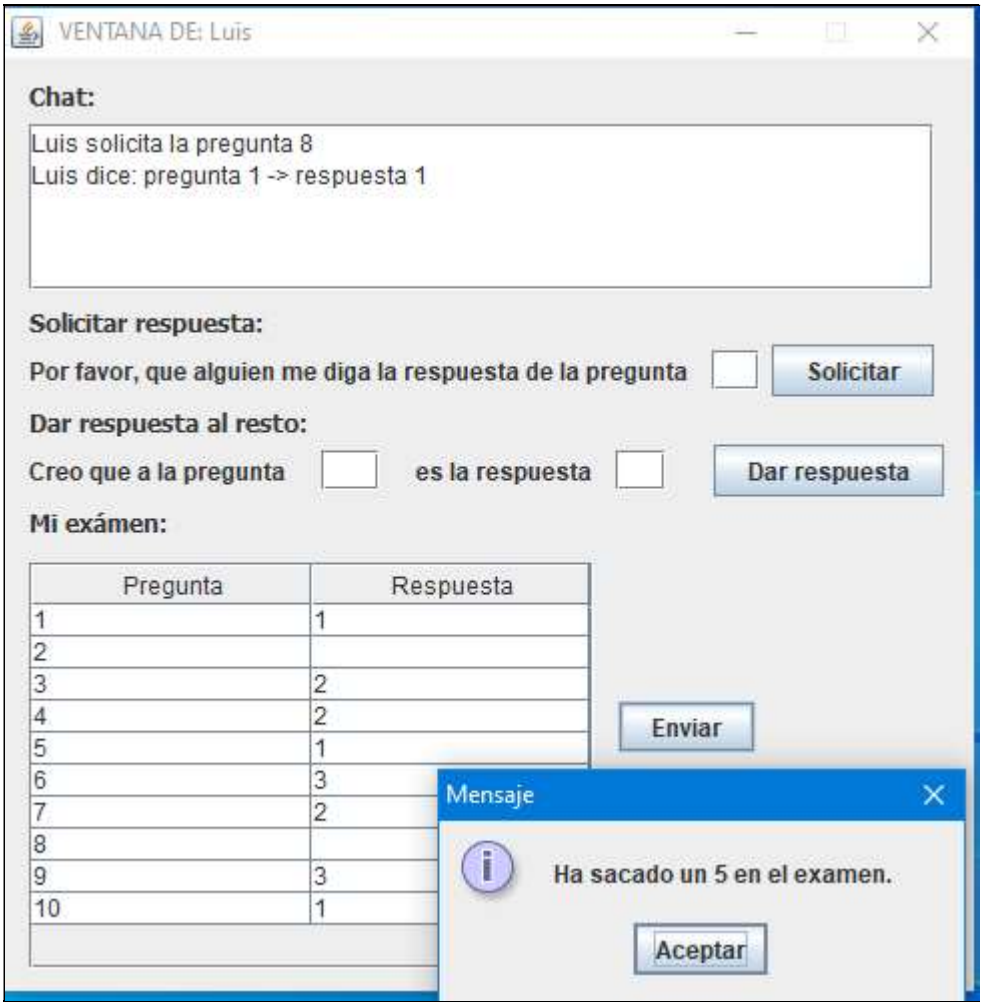

Ilustración 1 Ventana de Alumnos

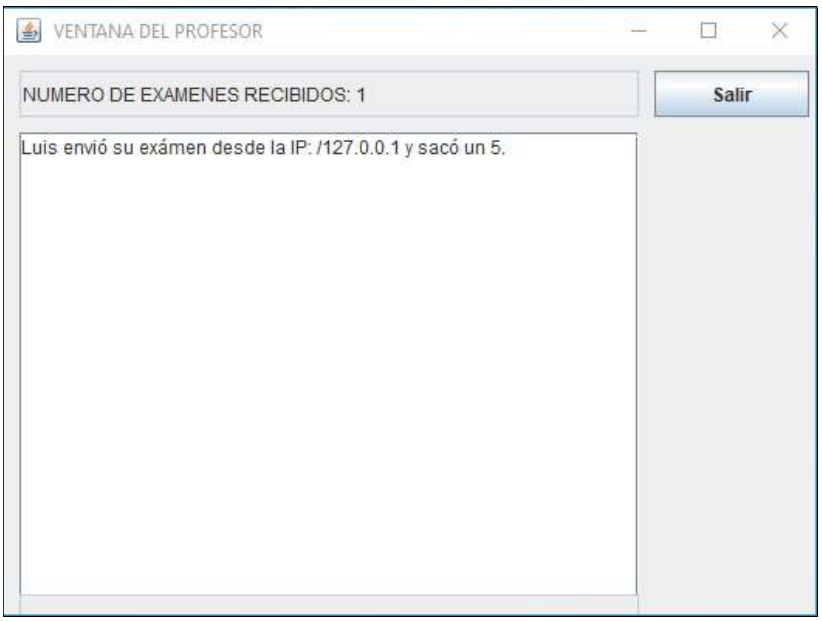

Ilustración 2 Ventana de Profesor