SEGUNDA PRUEBA PRÁCTICA Y ESCRITA 7 PLAZAS TÉCNICA/O AUXILIAR INFORMÁTICO (1 PLAZA RESERVADA PARA PERSONAS CON DISCAPACIDAD)

Conforme a lo establecido en las Bases Sexta y Séptima:

Segunda prueba práctica y escrita consistirá en la resolución de un supuesto práctico que será determinado por el tribunal inmediatamente antes de dar comienzo al ejercicio, y estará relacionado con el ejercicio de las funciones propias de la plaza convocada y/o con las materias establecidas en el *anexo II.*

En esta parte del ejercicio se evaluará la aplicación práctica de las materias contenidas en el *anexo II.*

El Tribunal de Selección determinará las condiciones de forma y estructura del supuesto práctico que se formule y deberá comunicarlo a las personas aspirantes antes de comenzar el ejercicio. Asimismo el Tribunal de Selección determinará el material complementario o bibliografía que en su caso, se permita utilizar durante la prueba, debiendo comunicarlo a las personas aspirantes con una antelación mínima de setenta y dos horas.

Para la realización conjunta de las dos pruebas las personas aspirantes dispondrán de un tiempo máximo de tres horas.

Segunda prueba.- Se calificara de 0 a 10 puntos, siendo preciso alcanzar una puntuación mínima de 5 puntos para superar la prueba.

A partir de la publicación en la página web municipal de la plantilla provisional de respuestas de la primera prueba, las personas aspirantes dispondrán de un plazo de 5 días naturales, a los efectos de presentar las alegaciones que estimen conveniente a la segunda prueba.

En el acta de la sesión o sesiones de calificación, y en las relaciones adjuntas a aquella se hará constar exclusivamente la calificación final que se adjudique a cada persona aspirante.

Zaragoza a 15 de mayo de 2018

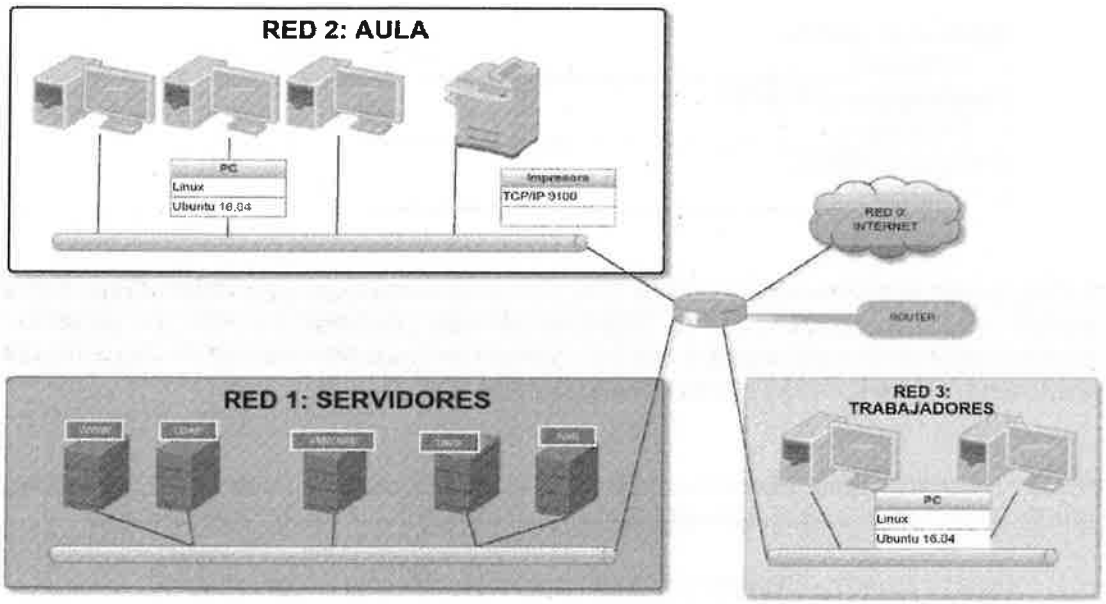
El Ayuntamiento de Zaragoza quiere poner en marcha un programa de formación para su personal. Para la gestión y puesta en marcha del proyecto se necesita realizar una serie de tareas informáticas entre las que se encuentran la programación del software de gestión y la administración de sistemas servidor y escritorio.

El entorno tecnológico se basa en:

- Software de gestión desarrollado en lenguaje Java y BBDD Oracle. Presentación y uso vía navegador web a través de servidor web Apache sobre sistema operativo Linux. Diseño visual basado en hojas de estilo en cascada CSS.
- Aula con 12 ordenadores de sobremesa con sistema operativo Ubuntu 16.04 conectados en red local con salida a Internet.

DIAGRAMA DE RED de base para todos los supuestos:

DIAGRAMA DE RED



RED 0: INTERNET
RED 1: SERVIDORES (NAS, DIRECTORIO, WEB, host Virtualización con VMWARE y DNS)
RED 2: AULA (PC alumnos e impresoras)
RED 3: TRABAJADORES
1 ROUTER de unión, SIN NAT excepto para la salida a Internet

Responda a las siguientes cuestiones relativas a este proyecto.

1) Proponga justificadamente un direccionamiento IPv4 para las zonas 1, 2 y 3 en función del número de equipos máximo que puede llegar a tener.

RED 1: Servidores – 25 servidores físicos y 225 virtuales
RED 2: AULA – 254 PC
RED 3: TRABAJADORES – 3000 PC

*nota: el direccionamiento debe permitir la conectividad entre las 4 zonas existentes.

2) Se requiere un dispositivo de red que permita:

- Limitar las conexiones TCP y UDP entre redes en base a dirección origen, dirección destino y puerto ó servicio.
- Limitar el ancho de banda de salida a Internet por puesto de la red
- Aplicar un control de contenidos (URL Filter) en el tráfico a Internet
- Realizar un análisis antivirus del tráfico
- Realizar una inspección profunda del tráfico de navegación cifrado por https

De los siguientes tipos de dispositivos, responda motivadamente cuál es el más indicado :

- **Router L3**: SI/NO.
  1. Motivos:_____
- **Cortafuegos L7**: SI/NO.
  1. Motivos:_____
- **Switch L2**: SI/NO.
  1. Motivos:_____

3) Proponga dos soluciones en los PCs Linux del aula que permitan disponer de acceso remoto seguro con el objetivo de dar soporte remoto al usuario y telemantenimiento del equipo. Deben usarse soluciones Software Libre locales al puesto (no basadas en servicios en la nube).

4) Se requiere que los PCs Linux del aula puedan usar una impresora multifunción conectada a su red local (impresión TCP/IP por puerto 9100).

4.1) Indique al menos 2 sistemas de impresión de Linux que lo permitan.

4.2) Usando el sistema estándar (por defecto) de impresión en el Linux del aula (Ubuntu 16.04), indique los comandos requeridos para las siguientes funciones:

- creación de impresoras
- borrado de impresoras
- gestión (monitorización y control) de las colas de impresión y los trabajos pendientes.

4.3) Describa las acciones necesarias para traducir los drivers de impresión disponibles al idioma local si no estuviesen disponibles por defecto.

5) Se necesita para un curso ejecutar en los Pcs Linux del aula una aplicación disponible solamente para SO Windows (ej. MS Access '97). Describa motivadamente al menos 2 soluciones a este problema (sin usar servicios externos a la red del Ayuntamiento).

6) Se necesita instalar software (LibreOffice en español) en los Pcs Linux del aula.

6.1) Indique al menos 2 formas para proveer (orígenes o fuentes) el software.

6.2) Indique los comandos requeridos sobre cada forma descrita para las siguientes funciones:

- instalar el software
- desinstalar el software incluyendo los ficheros de configuración
- verificar la instalación del software

7) Incluida, en el código de la aplicación, se dispone de la siguiente clase java:

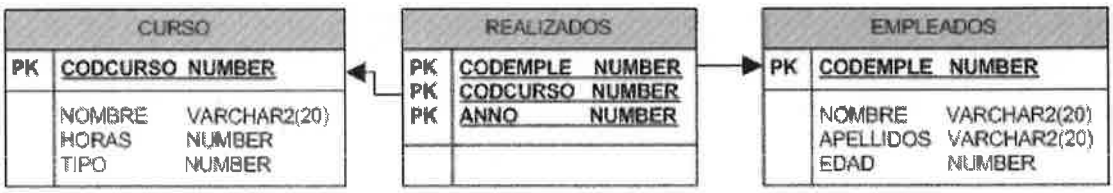
```
public class CursoBean  {

//Código del curso
private Long codCurso;

//nombre del curso
private String nombre;
        ....
        public void mostrarDatos() {
           System.out.println("Curso: "+getNombre());
         }
        ....
}
```

7.1) Definir dos métodos constructores de la clase, uno sin parámetros que por defecto asigne los valores nombre='curso1' y codCurso='1', y otro que pase por parámetro los valores para asignar a nombre y código del curso.

7.2) Definir una subclase pública CursoCompletoBean que herede de CursoBean. Debe tener dos atributos privados, uno que se llame **'valoracion'** tipo Long y otro llamado **'area'** tipo String. Definir los métodos get y set correspondientes al atributo **'valoracion'**.

7.3) Sobrescribir el método mostrarDatos en CursoCompletoBean, de manera que invoque al método mostrarDatos de CursoBean, y después muestre los valores de los atributos **'valoracion'** y **'area'**.

8) Dado el esquema de Base de Datos siguiente:



8.1) Definir una única sentencia SQL que seleccione todos los empleados que hayan realizado en total más de 15 horas de cursos entre los años 2015 y 2017 incluidos. Debe mostrarse el código y el apellido del empleado, y el total de horas de cursos realizados en esos años, ordenado por código de empleado ascendente.

8.2) Definir una única sentencia SQL que seleccione la lista de todos los empleados con todos los cursos que ha realizado en todos los años.
Mostrar el código y el apellido del empleado, el nombre del curso, el año de realización y las horas del curso, teniendo en cuenta que las horas de los cursos de tipo 1 se mostrarán multiplicadas por 2 y las de los cursos de tipo 2, multiplicadas por 3.
Deberán aparecer todos los empleados registrados aunque no hayan realizado ningún curso.
Ordenarlo por código de empleado, año del curso y nombre del curso.

9) Se define en java un método que accede a Base de Datos con una sentencia SQL y devuelve el resultado en un vector de objetos de tipo CursoBean, definido en el ejercicio 7.

Completar el código sustituyendo los círculos numerados por las partes de código que faltan y que se encuentran entre las opciones ofrecidas (no es necesario usar todas).

```
public List<① > listar(Long vCodEmple) {
    String sqls =
      "select distinct(c.codcurso), c.nombre" +
      " from realizados r, curso c" +
      " where codemple=" + vCodEmple+
      " and c.codcurso=r.codcurso" ;
      List<①> resultados = new ArrayList<①>();
      StringBuffer sql = new StringBuffer(sqls);
      Statement stmt = null;
      ResultSet rs = null;
      try {
         stmt = con.createStatement( );
         rs = stmt.executeQuery(sql.toString());
         ② (rs.③()) {
            CursoBean objeto = new CursoBean ();
            objeto.setCodCurso ( rs.④("codcurso") );
            objeto.setNombre ( rs.getString("nombre") );
            resultados.⑤(objeto);
         }
      } ⑥ (SQLException se) {
         System.out.println("Error");
      } finally {
         try {
            if (rs != null) rs.close();
            stmt.close();
         }⑥  (Exception mie) {
            mie.printStackTrace();
         }
      }
      return resultados;
}
```

Opciones ofrecidas:

notEnd	addElement	for	case
haveNext	insert	during	catch
next	append	while	capture
finalize	contains	lastIndexOf	size
long	add	until	throws
getLong	remove	if	throught
getLongValue	link	toString	CursoBean
getString	while	vCurso	CursoCompletoBean

10) En base a la estructura de tablas definidas en el ejercicio 8 con los siguientes datos almacenados:

Empleados:

CODEMPLE	NOMBRE	APELLIDOS	EDAD
1	pepe	garcia	30
2	juan	gomez	35
3	carlos	fernandez	40
4	javier	lopez	25

Curso:

CODCURSO	NOMBRE	HORAS	TIPO
1	ofimatica	10	1
2	atención público	20	2
3	admin. electrónica	30	1

Realizados:

CODEMPLE	CODCURSO	ANNO
1	2	2015
1	1	2017
1	1	2018
1	1	2015
2	1	2016
2	1	2018
3	2	2017

Se dispone del siguiente método en java:

```
public void listarCursos(Long vCodEmple) throws MiException {
    DAOFactory f = null;
    try
        f = new DAOFactory(Names.DATA_SOURCE);
        ConexionDAO dao = f.getConexionDAO();
//--cuerpo del programa
        List<CursoBean> lista= dao.listar(vCodEmple);
        StringBuffer mensaje;
        System.out.println("Cursos del Empleado "+vCodEmple);
        for (CursoBean curso : lista) {
            String textoFormateado = String.format("%-20s", curso.getCodCurso()+" "+
curso.getNombre());
            mensaje=new StringBuffer();
            mensaje.append(textoFormateado+" ");
            List<Long> listaAnnos=dao.listarAnnos(vCodEmple,curso.getCodCurso());
            for (Long vAnno : listaAnnos) {
               mensaje.append(" "+ vAnno);
            }
           System.out.println(mensaje);
        }
//--fin del cuerpo del programa
    } ...
   }
```

Donde CursoBean es la clase descrita en el ejercicio 7.

El método listar, es el método descrito en el ejercicio 9.

El método listarAnnos se define como
public List<Long> listarAnnos(Long vCodEmple, Long vCurso){..}
y devuelve la lista de años que devuelve el select definido en el siguiente String:
String sqls ="select anno from realizados where codemple=" +
vCodEmple+ " and codcurso=" + vCurso+ " order by anno " ;

Dar por hecho que, aunque no se muestre, la clase contiene el código necesario para que todo funcione correctamente (conexión a BD, importación de librerías, declaración de excepciones, etc.).
En la BD solo se guardan datos de los últimos 4 años.

10.1) Mostrar la salida por pantalla del programa listarCursos al invocarlo con la llamada:
listarCursos(1L);

10.2) Reescribir el código del cuerpo que sea necesario para que se muestre el resultado en forma de tabla, con el siguiente formato:

Empleado 1	2015	2016	2017	2018
1 jardineria	x	_	x	_
2 bricolaje	_	x	_	_
.....				

'x'-curso realizado
'_'-curso no realizado

Si lo considera necesario, en el desarrollo de la solución puede utilizar el siguiente código:

```
//Definición de un array
tipo_dato[] nombre_array = new tipo_dato[tamaño];

//Obtener el año actual (devuelve un tipo int):
Calendar.getInstance().get(Calendar.YEAR)

//cast de int a Long
new Long(integer);
```

11) Dada una página html, con las siguientes etiquetas:

```
<html>
<head>
.....
<link rel="stylesheet" href="./styles.css" type="text/css" />
</head>
<body>
<p>Apartado 1. </p>
<p class="nuevo">Apartado 2. </p>
<div>
        <p class="nuevo">Apartado 3. </p>
        <p>Apartado 4. </p>
</div>
<div class="viejo">
        Apartado 5.
        <p class="nuevo">Apartado 6. </p>
        <p>Apartado 7. </p>
</div>
<p class="nuevo">Apartado 8. </p>
<div id="viejo">
        <p>Apartado 9. </p>
</div>
<div>
        <p>Apartado 10. </p>
</div>
</body>
</html>
```

y dado el fichero styles.css:

```
p {color: black;}
p.nuevo {color: blue;}
p#nuevo {color:green;}
div {color:#9900FF;}
div.viejo {border: black 3px solid;}
div p.nuevo {color: #FFCC33;}
div.viejo p.nuevo {color:#66FF66;}
div#viejo p {color: #FF00FF;}
```

Indicar el color con el que se mostrará cada apartado (código o nombre del color) y si lleva borde o no.

12) Describa el objetivo global del *script* que se detalla a continuación (/servicios/scriptOp1.sh), así como una explicación detallada de las siguientes líneas de su código:

12.1) líneas de la 9 hasta la 14.
12.2) líneas 15 y 16.
12.3) líneas de la 29 hasta la 35.
12.4) línea 38.
12.5) líneas de la 44 hasta la 59.
12.6) líneas 60 y 67.
12.7) líneas 41, 42 y 69.

```
1   #!/bin/bash
2   ############################## VARIABLES ###################################
3   export A_L="/almacen/L"
4   if [[ ! -d ${A_L}/EJEMPLO ]]
5    then
6     echo "ERROR, -NO- existe el directorio ORIGEN"
7     exit 1
8   fi
9   export M="$(echo ${HOSTNAME}|/usr/bin/awk -F\. '{print $1}')"
10  export B_L="/NAS/almacen/L.${M}"
11  if [[ ! -d ${B_L}/EJEMPLO ]]
12   then
13    mkdir -pv ${B_L}/EJEMPLO
14  fi
15  export FECHA=$(date +%Y%m%d-%H%M)
16  export L=${A_L}/RyS/$(basename ${0}).${FECHA}.log
17  if [[ ! -d ${A_L}/RyS ]]
18   then
19    mkdir -pv ${A_L}/RyS
20  fi
21  ############################## PROCESOS ####################################
22  proceso_A()
23  {
24   for i in $(ls *.log)
25    do
26     mv -v ${i} ${B_L}/EJEMPLO/${i}.${FECHA}
27   done
28  }
29  proceso_B()
30  {
31   for i in $(ls ${B_L}/EJEMPLO/*|grep -v '.gz')
32    do
33     gzip -f9v ${i}
34   done
35  }
36  proceso_C()
37  {
38   find . -name "*.gz" -mtime +180 -exec rm -fv {} \;
39  }
40  ############################## MAIN ########################################
41  for proceso in 1
42   do
43    ACCION="A B C"
44    until [[ -z ${1} ]]
45     do
46      case ${1} in
47       "-p") shift
48             ACCION=$(echo ${1}|tr '[a-z]' '[A-Z]')
49             if [[ ${ACCION} = A || ${ACCION} = B || ${ACCION} = C ]]
50              then
51              shift
52             else
53              echo "ERROR en proceso tecleado: -${1}-"
54              exit 1
55             fi;;
```

```
56              *) echo "ERROR, se ha tecleado un parámetro NO válido -${1}"
57                 exit 1;;
58            esac
59          done
60          pushd ${A_L}/EJEMPLO
61          for x in $(echo ${ACCION})
62           do
63            echo "PROCESO: ${x}"
64            proceso_${x} 2>&1
65            echo ""
66          done
67          popd
68          echo "SCRIPT $(basename ${0}) TERMINADO"
69        done 2>&1|tee ${L}
```

13) El contenido del archivo *crontab* del usuario *root* en nuestro sistema es el siguiente:

```
1)  0,15,30,45 * * * * systemctl reload-or-try-restart apache.service
>>/var/log/apache.txt
2)  00 03 * 1-15 * operador -c "/servicios/scriptOp1.sh" /dev/null 2>&1
3)  */15 * * * * /usr/bin/who >> /var/log/usuarios.txt
4)  00 03 1,15 * * su - operador -c "/servicios/scriptOp1.sh" > /dev/null 2>&1
5)  23 0-23/2 * * 0 cd /root/isos;wget
http://example.com/fichero_a_descargar.iso
```

Conteste, explicando de manera justificada, a las siguiente cuestiones:

13.1) ¿Cuál de las líneas de este *crontab* nos permite la ejecución del *script* del ejercicio 12 por parte del usuario *operador*, los días 1 y 15 de cada mes a las 3 de la madrugada?
13.2) ¿Qué hacen el resto de líneas del *crontab*?

14) En el archivo *httpd.conf* del Servidor web Apache, hay definido un *VirtualHost* para la publicación de los ficheros de log de las aplicaciones de Gestión:

```
1)    <VirtualHost *:80>
2)          ServerName ejemplo.red.zaragoza.es
3)          DocumentRoot "/almacen/log/"
4)          Alias /log/ "/almacen/log/"
5)          <Directory "/almacen/log/">
6)                Options FollowSymLinks MultiViews Indexes
7)                IndexOptions FancyIndexing FoldersFirst NameWidth=*
8)                AllowOverride AuthConfig
9)                AuthType Basic
10)               AuthName "Logs de Gestión"
11)               AuthUserFile /etc/httpd/conf.d/.htpasswd_ejemplo
12)               Require valid-user
13)         </Directory>
14)   </VirtualHost>
```

Explicar detalladamente cada línea de este *VirtualHost*.