ACTA 12/02/19
ANEXO.

OFERTA DE EMPLEO PÚBLICO 2014-2015-2016
PROGRAMADOR/A INFORMÁTICO/A
2º EJERCICIO
12/02/2019

Una Administración Pública organiza cursos para la formación continua de sus empleados/as y a los que también pueden asistir personas de otras administraciones públicas.

Decide poner en marcha una serie de servicios para facilitar la planificación y gestión de esos cursos (las solicitudes de inscripción, las listas de admitidos/excluidos/en reserva, etc) y facilitar que las personas interesadas se puedan apuntar a los cursos ofertados y consultar el estado de sus solicitudes.

Para ello se ha creado una base de datos de partida, de nombre **Bdformacion**, con la siguiente colección de tablas básicas:

cursos	
cur_codigo	Integer identity *(Primary Key)*
cur_titulo	varchar(150)
cur_fecini	date
cur_fecfin	date
cur_publicar	Boolean
cur_modalidad	Integer
cur_plazas	Integer *default 0*
cur_finplazosol	date
cur_horastotal	Integer *default 0*
cur_horaspresen	Integer *default 0*
cur_estado	Integer *default 0*
cur_condiploma	Boolean default false
cur_temario	blob

cur_modalidad	
cmo_codigo	Smallint *(Primary Key)*
cmo_descri	varchar(50)

cur_estados	
ces_codigo	Smallint *(Primary Key)*
ces_grupo	char(10)
ces_descri	varchar(50)

cur_solicitud	
cso_codigo	Integer identity *(Primary Key)*
cso_curso	integer
cso_alumno	integer
cso_fecsolicita	date
cso_estadosol	Integer *default 10*
cso_ordensol	Integer default 0
cso_confiralum	Boolean default false

personas	
perso_codigo	Integer identity *(Primary key)*
perso_relaempre	smallint
perso_nif	char(12)
perso_nombre	varchar(80)
perso_apellid1	varchar(80)
perso_apellid2	varchar(80)

perso_relaempre	
prel_codigo	smallint *(Primary key)*
prel_descripcio	smallint

Valores en tablas auxiliares:

cur_modalidad	
0	Presencial
1	Semipresencial
2	A distancia
3	Otros

perso_relaempre	
0	Externa
1	Interna (personal contratado)

cur_estados		
0	cursos	En elaboración
1	cursos	Ofertado
2	cursos	Selección
3	cursos	Cerrado
5	cursos	Finalizado
6	cursos	Suspendido
7	cursos	Aplazado
10	solicita	Pendiente
11	solicita	Admitida
12	solicita	Excluída
13	solicita	En reserva
15	solicita	Invitado/a
16	solicita	Renuncia
19	solicita	Otros

En el escenario descrito, resuelva las siguientes tareas:

1. Diagrama de las relaciones entre las tablas. (1 punto)

2. Escribir las instrucciones SQL para realizar las siguientes modificaciones en la estructura de la base de datos : (2 puntos)
   a) Crear un índice de nombre **ix_solalumno** sobre la tabla *cur_solicitud* de forma que se impida que cada alumno realice más de una solicitud por curso
   b) Añadir a la tabla *personas* un campo de nombre **perso_correo** en cual guardar el valor de la cuenta de correo electrónico de la persona

3. Escribir las consultas SQL de selección/alteración de datos para: (2 puntos)
   a) Obtener la información de los cursos en estado Ofertado y activados para publicar
   b) Marcar como "Admitida" todas las solicitudes del curso 25 cuyo orden no supere el número de plazas del curso

4. Escribir un formulario HTML5 que permita a cada persona solicitar darse de alta en un curso. (2 puntos)
   a) El formulario debe consistir en los siguientes elementos:
      1. Una cabecera.
      2. Campos que permitan a la persona escribir sus datos.
      3. Campo que permita seleccionar si la persona está contratada por la Administración convocante o es personal externo. Por defecto debe estar seleccionada la opción de persona contratada (interna).
      4. Campo que permita elegir el código del curso al que se desea apuntar. Debe poder elegir entre 5 cursos posibles (Curso 1, Curso 2, Curso 3, Curso 4, Curso 5).
   b) Diseñar una hoja de estilos de nombre estilos.css, que permita:
      1. Centrar la cabecera del formulario.
      2. Mostrar los campos de entrada de datos en disposición vertical (uno encima de otro).
      3. El color de fondo del campo que permite escribir el NIF de la persona en amarillo y la letra en negrita.
      4. Indicar como y donde se referencia a dicha hoja de estilo en el fichero HTML.

5. Escribir código php para obtener una función **DameDiploma** que, recibiendo como parámetros el código de un curso y un NIF, devuelva el fichero del diploma, si existe, y en caso contrario: que devuelva FALSE y saque en pantalla un mensaje informando de ello.
   Tal fichero se supone que está en la ruta '/home/tmp/' del servidor web y que su nombre tiene el siguiente formato: `Diplom_025_0000000T.pdf`, donde 25 es el código del curso y 0000000T el nif del alumno. (1 punto)

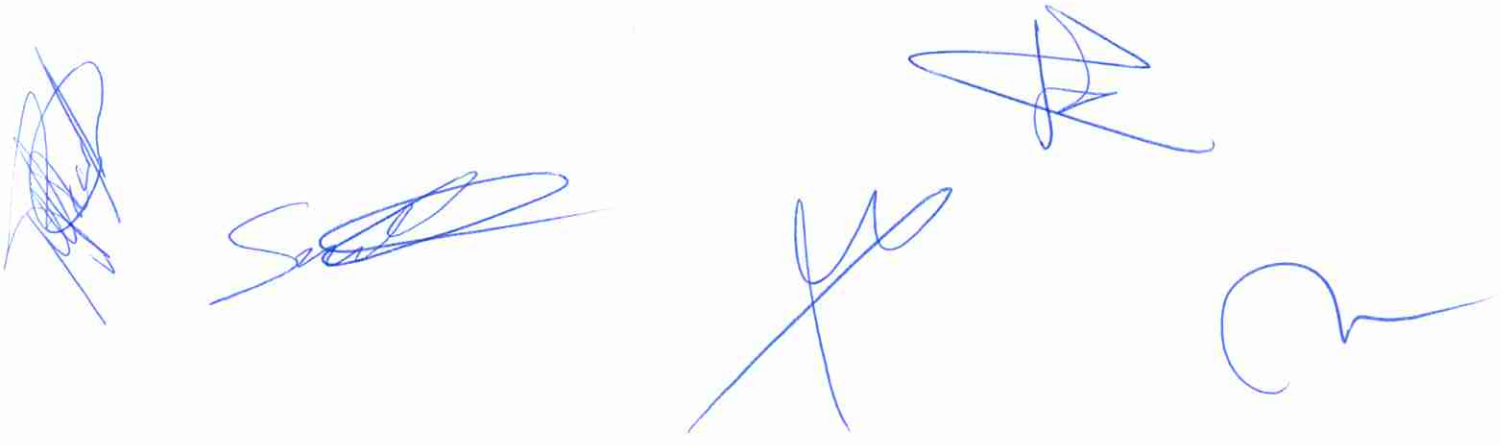
6. Dadas las clases java Curso, EstadoPersonaCurso y MailManager: (2 puntos)

   a) Escribir una función **searchPersonasBBDD** dentro de la clase *Curso* cuya funcionalidad sea consultar los siguientes datos: código de persona, correo electrónico de la persona, código del curso solicitado, título del curso solicitado y estado de la solicitud. Con dichos datos se debe rellenar el Vector llamado *listaPersonas*.

   b) Escribir una función **enviarCorreo** dentro de la clase *Curso*, cuyo parámetro de entrada sea el Vector *listaPersonas* que contiene la lista de personas con el estado de solicitud de cada curso solicitado y cuya funcionalidad sea enviar un correo electrónico a través de la clase *MailManager* a cada una de las cuentas, con el asunto 'Respuesta a su solicitud al curso XXX' y como cuerpo del texto: 'Su solicitud al curso XXX ha sido ZZZ', donde XXX es el título del curso y ZZZ es el estado recibido como parámetro.

Curso.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Vector;

public class Curso {

   private static Vector<EstadoPersonaCurso> listaPersonas = new Vector<EstadoPersonaCurso>();

   private static Connection conexion = null;

   public Curso() {
      super();
      initBBDD();
   }

   private static void initBBDD() {
      try {
         Connection conexion = DriverManager.getConnection("jdbc:mysql://servidor:3306/database",
            "usuario", "password");
         conexion.setAutoCommit(false);
      }
      catch(Exception exc) {
         exc.printStackTrace();
      }
   }

   public static void main(String[] args) {
      Curso ec = new Curso();


   }
}
```



MailManager.java

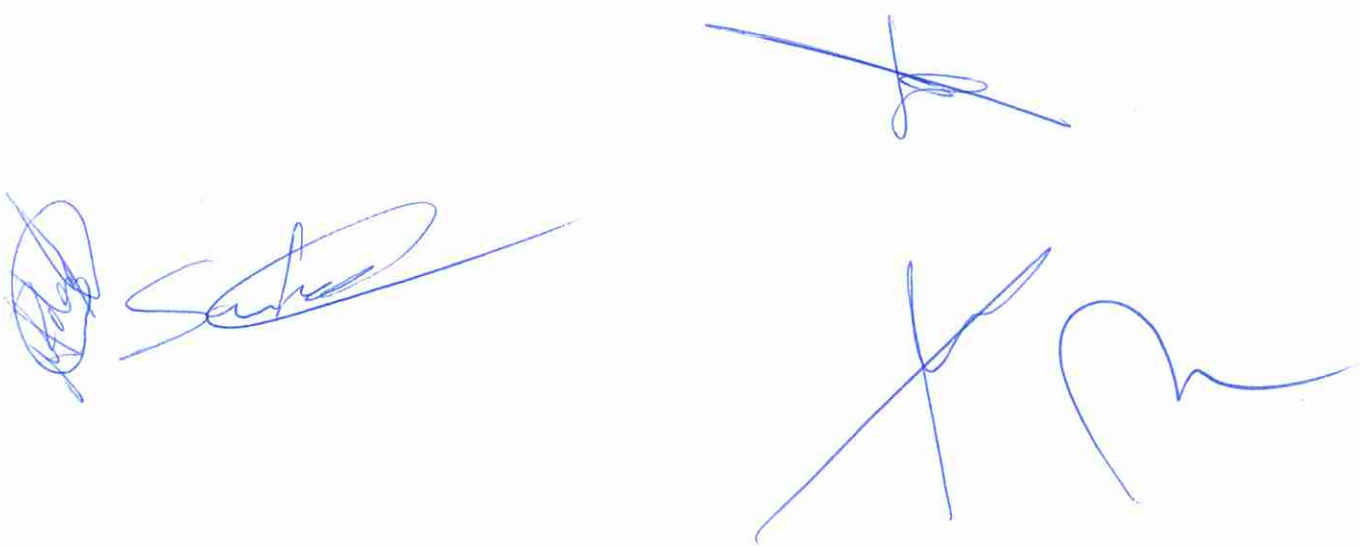
```
import java.util.Properties;
import javax.mail.MessagingException;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class MailManager {
   private final Properties properties = new Properties();

   private String password;

   private Session session;

   private void init() {
      properties.put("mail.smtp.host", "mail.gmail.com");
      properties.put("mail.smtp.starttls.enable", "true");
      properties.put("mail.smtp.port",25);
      properties.put("mail.smtp.mail.sender","emisor@gmail.com");
      properties.put("mail.smtp.user", "usuario");
      properties.put("mail.smtp.auth", "true");

      session = Session.getDefaultInstance(properties);
   }

   public void enviarMail(String direccionDestino, String asunto, String texto) {
      init();
      try{
         MimeMessage message = new MimeMessage(session);
         message.setFrom(new InternetAddress((String)properties.get("mail.smtp.mail.sender")));

         message.addRecipient(Message.RecipientType.TO, new InternetAddress(direccionDestino));
         message.setSubject(asunto);
         message.setText(texto);

         Transport t = session.getTransport("smtp");
         t.connect((String)properties.get("mail.smtp.host"), (String)properties.get("mail.smtp.user"),
                        "password");
         t.sendMessage(message, message.getAllRecipients());
         t.close();
      }
      catch (MessagingException me){
         return;
      }
   }
}
```

EstadoPersonaCurso.java

```
import java.util.Enumeration;
import java.util.Hashtable;

public class EstadoPersonaCurso {

   private int codCurso;
```

```
private String tituloCurso;

private int codPersona;

private String email;

private String estado;

public EstadoPersonaCurso() {
      super();
}

public EstadoPersonaCurso(int codPersona, String email, int codCurso, String descCurso,
                String   estado) {
   super();
   init(codPersona, email, codCurso, descCurso, estado);
}

public void init(int codPersona, String email, int codCurso, String descCurso, String estado) {
   codPersona(codPersona);
   if(email!=null)
      setEmail(email);
   setCodigoCurso(codCurso);
   if(descCurso!=null)
      setTituloCurso(descCurso);
   if(estado!=null)
      setEstadoSolicitud(estado);
}

public void codPersona(int codigoPersona){
      codPersona = codigoPersona;
}

public int godPersona() {
   return codPersona;
}

public void setEmail(String direccionEmail){
      email = direccionEmail;
}

public String getEmail() {
   return email;
}

public void setCodigoCurso(int codigo){
      codCurso = codigo;
}

public int getCodigoCurso() {
   return codCurso;
}

public void setTituloCurso(String descCurso){
      tituloCurso = descCurso;
}

public String getTituloCurso() {
   return tituloCurso;
```

```
    }

    public void setEstadoSolicitud(String estadoCurso){
          estado = estadoCurso;
    }

    public String getEstadoSolicitud() {
       return estado;
    }
}
```



OFERTA DE EMPREGO PÚBLICO 2014-2015-2016
PROGRAMADOR/A INFORMÁTICO/A
2º EXERCICIO
12/02/2019

Unha Administración Pública organiza cursos para a formación continua dos seus empregados/as e aos que tamén poidan asistir persoas de outras administracións públicas.

Decide poñer en marcha unha serie de servizos para facilitar a planificación e xestión deses cursos (as solicitudes de inscrición, as listas de admitidos/excluídos/en reserva, etc) e facilitar que as persoas interesadas se poidan apuntar aos cursos ofertados e consultar o estado das súas solicitudes.

Para elo creouse unha base de datos de partida, de nome `Bdformacion`, coa seguinte colección de táboas básicas:

cursos	
cur_codigo	Integer identity *(Primary Key)*
cur_titulo	varchar(150)
cur_fecini	date
cur_fecfin	date
cur_publicar	Boolean
cur_modalidad	Integer
cur_plazas	Integer *default 0*
cur_finplazosol	date
cur_horastotal	Integer *default 0*
cur_horaspresen	Integer *default 0*
cur_estado	Integer *default 0*
cur_condiploma	Boolean default false
cur_temario	blob

cur_solicitud	
cso_codigo	Integer identity *(Primary Key)*
cso_curso	integer
cso_alumno	integer
cso_fecsolicita	date
cso_estadosol	Integer *default 10*
cso_ordensol	Integer default 0
cso_confiralum	Boolean default false

cur_modalidad	
cmo_codigo	Smallint *(Primary Key)*
cmo_descri	varchar(50)

personas	
perso_codigo	Integer identity *(Primary key)*
perso_relaempre	smallint
perso_nif	char(12)
perso_nombre	varchar(80)
perso_apellid1	varchar(80)
perso_apellid2	varchar(80)

cur_estados	
ces_codigo	Smallint *(Primary Key)*
ces_grupo	char(10)
ces_descri	varchar(50)

perso_relaempre	
prel_codigo	smallint *(Primary key)*
prel_descripcio	smallint

No escenario descrito, resolva as seguintes tarefas:

1. Diagrama das relacións entre las táboas. (1 punto)

2. Escribir as instrucións SQL para realizar as seguintes modificacións na estrutura da base de datos: (2 puntos)
   a) Crear un índice de nome **ix_solalumno** sobre a táboa *cur_solicitud* de forma que se impida que cada alumno realice mais de unha solicitude por curso
   b) Engadir á táboa *personas* un campo de nome **perso_correo** no cal gardar o valor da conta de correo electrónico da persoa

3. Escribir as consultas SQL de selección/alteración de datos para: (2 puntos)
   a) Obter a información dos cursos en estado Ofertado e activados para publicar
   b) Marcar como "Admitida" todas as solicitudes do curso 25 cuxo orde non supere o número de prazas do curso

4. Escribir un formulario HTML5 que permita a cada persoa solicitar darse de alta nun curso. (2 puntos)
   a) O formulario debe consistir nos seguintes elementos:
      1. Unha cabeceira.
      2. Campos que permitan á persoa escribir os seus datos.
      3. Campo que permita seleccionar se a persoa está contratada pola Administración convocante ou é persoal externo. Por defecto debe estar seleccionada a opción de persoa contratada (interna).
      4. Campo que permita elixir o código do curso ao que se desexa apuntar. Debe poder elixir entre 5 cursos posibles (Curso 1, Curso 2, Curso 3, Curso 4, Curso 5).
   b) Deseñar unha folla de estilos de nome estilos.css, que permita:
      1. Centrar a cabeceira do formulario.
      2. Mostrar os campos de entrada de datos en disposición vertical (un enriba do outro).
      3. O color de fondo do campo que permite escribir o NIF da persoa en amarelo e a letra en negriña.
      4. Indicar como e onde se referencia a dita folla de estilo no ficheiro HTML.

5. Escribir código php para obter unha función **DameDiploma** que, recibindo como parámetros o código dun curso e un NIF, devolva o ficheiro do diploma, se existe, e en caso contrario: que devolva FALSE e saque en pantalla unha mensaxe informando delo.
   Tal ficheiro se supón que está na ruta '/home/tmp/' do servidor web e que o seu nome ten o seguinte formato: `Diplom_025_0000000T.pdf`, onde 25 é o código do curso e 0000000T o nif do alumno. (1 punto)

6. Dadas as clases java Curso, EstadoPersonaCurso e MailManager: (2 puntos)

   a) Escribir unha función **searchPersonasBBDD** dentro da clase *Curso* cuxa funcionalidade sexa consultar os seguintes datos: código de persoa, correo electrónico da persoa, código do curso solicitado, título do curso solicitado e estado da solicitude. Con ditos datos se debe rechear o Vector chamado *listaPersonas*.

   b) Escribir unha función **enviarCorreo** dentro da clase *Curso*, cuxo parámetro de entrada sexa o Vector *listaPersonas* que contén a lista de persoas co estado de solicitude de cada curso solicitado e cuxa funcionalidade sexa enviar un correo electrónico a través da clase *MailManager* a cada unha das contas, co asunto 'Respuesta a su solicitud al curso XXX' e como corpo do texto: 'Su solicitud al curso XXX ha sido ZZZ', onde XXX é o título do curso e ZZZ é o estado recibido como parámetro.

Curso.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Vector;

public class Curso {

    private static Vector<EstadoPersonaCurso> listaPersonas = new Vector<EstadoPersonaCurso>();

    private static Connection conexion = null;

    public Curso() {
        super();
        initBBDD();
    }

    private static void initBBDD() {
        try {
            Connection conexion = DriverManager.getConnection("jdbc:mysql://servidor:3306/database",
                "usuario", "password");
            conexion.setAutoCommit(false);
        }
        catch(Exception exc) {
            exc.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Curso ec = new Curso();


    }
}
```

MailManager.java

```
import java.util.Properties;
import javax.mail.MessagingException;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class MailManager {
   private final Properties properties = new Properties();

   private String password;

   private Session session;

   private void init() {
      properties.put("mail.smtp.host", "mail.gmail.com");
      properties.put("mail.smtp.starttls.enable", "true");
      properties.put("mail.smtp.port",25);
      properties.put("mail.smtp.mail.sender","emisor@gmail.com");
      properties.put("mail.smtp.user", "usuario");
      properties.put("mail.smtp.auth", "true");

      session = Session.getDefaultInstance(properties);
   }

   public void enviarMail(String direccionDestino, String asunto, String texto) {
      init();
      try{
         MimeMessage message = new MimeMessage(session);
         message.setFrom(new InternetAddress((String)properties.get("mail.smtp.mail.sender")));

         message.addRecipient(Message.RecipientType.TO, new InternetAddress(direccionDestino));
         message.setSubject(asunto);
         message.setText(texto);

         Transport t = session.getTransport("smtp");
         t.connect((String)properties.get("mail.smtp.host"), (String)properties.get("mail.smtp.user"),
                        "password");
         t.sendMessage(message, message.getAllRecipients());
         t.close();
      }
      catch (MessagingException me){
         return;
      }
   }
}
```

EstadoPersonaCurso.java

```
import java.util.Enumeration;
import java.util.Hashtable;

public class EstadoPersonaCurso {

   private int codCurso;

   private String tituloCurso;

   private int codPersona;

   private String email;

   private String estado;

   public EstadoPersonaCurso() {
         super();
   }

   public EstadoPersonaCurso(int codPersona, String email, int codCurso, String descCurso,
                  String  estado) {
      super();
      init(codPersona, email, codCurso, descCurso, estado);
   }

   public void init(int codPersona, String email, int codCurso, String descCurso, String estado) {
      codPersona(codPersona);
      if(email!=null)
         setEmail(email);
      setCodigoCurso(codCurso);
      if(descCurso!=null)
         setTituloCurso(descCurso);
      if(estado!=null)
         setEstadoSolicitud(estado);
   }

   public void codPersona(int codigoPersona){
         codPersona = codigoPersona;
   }

   public int godPersona() {
      return codPersona;
   }

   public void setEmail(String direccionEmail){
         email = direccionEmail;
   }

   public String getEmail() {
      return email;
   }

   public void setCodigoCurso(int codigo){
         codCurso = codigo;
   }
```

```
    public int getCodigoCurso() {
        return codCurso;
    }

    public void setTituloCurso(String descCurso){
            tituloCurso = descCurso;
    }

    public String getTituloCurso() {
        return tituloCurso;
    }

    public void setEstadoSolicitud(String estadoCurso){
            estado = estadoCurso;
    }

    public String getEstadoSolicitud() {
        return estado;
    }
}
```