PRUEBAS SELECTIVAS PARA EL INGRESO EN LA ESCALA TÉCNICA AUXILIAR DE INFORMÁTICA, GRUPO C, SUBGRUPO C1

(Resolución Rectoral de 7 de febrero de 2023)

Parte 2: Supuesto Práctico Bloque III - Desarrollo

17 de junio de 2023

[Cada pregunta vale un punto]

Una universidad pública española aprueba en su Consejo de Gobierno la posibilidad de realizar voto por correo en las elecciones a Rector. Una vez aprobado debe desarrollar la aplicación que permita gestionar todo el proceso, desde la solicitud hasta el momento de entrega en la mesa correspondiente de los votos recibidos. Se asigna a la unidad de informática de la universidad el desarrollo de la aplicación para gestionar todo el proceso.

El modelo de datos que desarrolla la unidad, basándose en los requisitos presentados por la Secretaría General, es el siguiente:

```
Persona (DNI, NRP, Nombre, Apellidos);
DatosLaborales (NRP, Categoría, Centro);
DatosVotacion(NRP, Mesa, Centro);
SolicitudesVotoPorCorreo (DNI, Fecha, Estado(rechazado;aceptado;espera));
RecibidosVotosPorCorreo (DNI, Fecha);
EntregadosVotosPorCorreo (DNI, Mesa, Fecha);
```

(* NRP: Número de Registro de Personal, es un identificador único del trabajador dentro de la organización).

1) Defina el concepto de herencia en un lenguaje orientado a objetos y ponga un ejemplo de código de herencia de clases en PHP o JAVA, con las entidades que aparecen en el modelo de datos del enunciado. Justifique la elección de las entidades.

2) Defina un conjunto de clases que permita, definiendo un objeto, obtener a través de sus métodos:

a) Si un trabajador ha solicitado el voto por correo.

b) Si se le ha concedido el voto por correo.

c) Si ha sido recibido el voto por correo.

d) Si se ha entregado el voto a la mesa correspondiente.

Codifique una breve aplicación que utilice las clases anteriores para sacar por consola la información solicitada en cada uno de los apartados. Puede utilizar lenguaje PHP o JAVA.

3) Amplíe el modelo de datos propuesto para poder gestionar el escrutinio del voto por correo, teniendo en cuenta que cada mesa se compone de urnas, y que éstas corresponden a una categoría laboral concreta. Existen 4 categorías laborales, cuya ponderación o peso en la votación final se refleja en la siguiente tabla:

Categoría laboral	Ponderación voto
Profesores doctores con vinculación permanente	51%
Resto del profesorado y personal investigador	16%
Estudiantes	24%
Personal de administración y servicios	9%

4) Implemente los métodos que considere necesarios, en las clases que considere oportunas, para realizar el cálculo del escrutinio según se van recibiendo los resultados de cada urna, teniendo en cuenta las ponderaciones y el modelo de la pregunta anterior.

5) La aplicación requiere desarrollar una API RESTful, para implementar las funcionalidades mediante llamadas desde la capa de presentación. En concreto se pide poder realizar las siguientes operaciones:

- Asignar el centro y mesa en que tiene que votar un trabajador.
- Saber si una persona ha pedido el voto por correo.
- Actualizar la información en la BBDD del estado de la petición del voto por correo.

a) Defina la interfaz de los métodos necesarios para estas operaciones.

b) Implemente estos métodos en el lenguaje de programación de su elección (JAVA o PHP).

6) Se implementará un método adicional que devuelva, en formato JSON, toda la información existente sobre un trabajador (datos personales y datos laborales).

a) Indique una posible estructura de la respuesta JSON, teniendo en cuenta las entidades existentes.

b) Defina la estructura de un objeto para almacenar el JSON recibido.

7) Defina las características de una API tipo RESTful. Defina las características de un servicio de tipo SOAP. Señale las diferencias entre ambas.

8) Indique al menos tres diferencias entre la utilización de un ORM como Hibernate o Doctrine a no utilizarlo en el acceso a una base de datos.

9) Dado el siguiente código Java con Hibernate generado para la tabla Persona (puede considerar una estructura equivalente en código PHP con Doctrine):

```
import javax.persistence.*;
@Entity
@Table(name = "Persona")
public class Persona {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(name = "DNI")
   private String dni;

   @Column(name = "NRP")
   private String nrp;

   @Column(name = "Nombre")
   private String nombre;

   @Column(name = "Apellidos")
   private String apellidos;

   // Constructor, getters y setters
   ...
}
```

a) Explique brevemente cómo se crearía la tabla Persona.
b) Explique brevemente para qué sirven cada una de las anotaciones utilizadas: @Entity, @Table, @Id, @GeneratedValue y @Column.

10) Asuma que la clase de entidad Persona está correctamente mapeada en un ORM según la declaración realizada en el apartado anterior. Tome como referencia el siguiente código para implementar un método llamado insertarPersona que recibe como parámetros el DNI, el NRP, el nombre y los apellidos de una persona y los inserte en la tabla Persona de la base de datos.
Se pide completar la implementación en Java con Hibernate (o en PHP con Doctrine) para realizar la inserción en la base de datos.

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import paquete.clase.Persona;
import paquete.util.HibernateUtil;

public class PersonaDAO {

   public void insertarPersona(String dni, String nrp, String nombre, String apellidos) {
      SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

      try (Session session = sessionFactory.openSession()) {
         Transaction transaction = session.beginTransaction();

         // Aquí irá el Código que se pide implementar

         transaction.commit();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

}
```