PRÁCTICA TRONCAL (versión 1)

Este documento describe la práctica troncal de la asignatura de Ingeniería del Software III que deberán realizar los alumnos, preferentemente en parejas. La práctica completa está valorada con dos puntos y deberá ser entregada a los profesores de la asignatura en el mes de Enero. Ésta será evaluada mediante un examen práctico, que deberá ser resuelto individualmente, valorado a su vez en otros cuatro puntos, y que consistirá en la realización de una modificación sobre la práctica de cada alumno.

Este documento se trata de la versión 1, ya que describe únicamente la primera fase o hito de la práctica: la capa de persistencia.

1. CAPA DE PERSISTENCIA

Esta capa albergará el modelo de objetos y los datos pertenecientes al dominio de una entidad docente (a saber: asignaturas y unidades, profesores, alumnos, evaluaciones, etc.). Se deberá implementar haciendo uso del framework Hibernate y la base de datos MySQL.

1.1. Arquitectura

La Figura 1 muestra la arquitectura de la capa de persistencia y el lugar que ocupa en la arquitectura por capas general de la práctica.

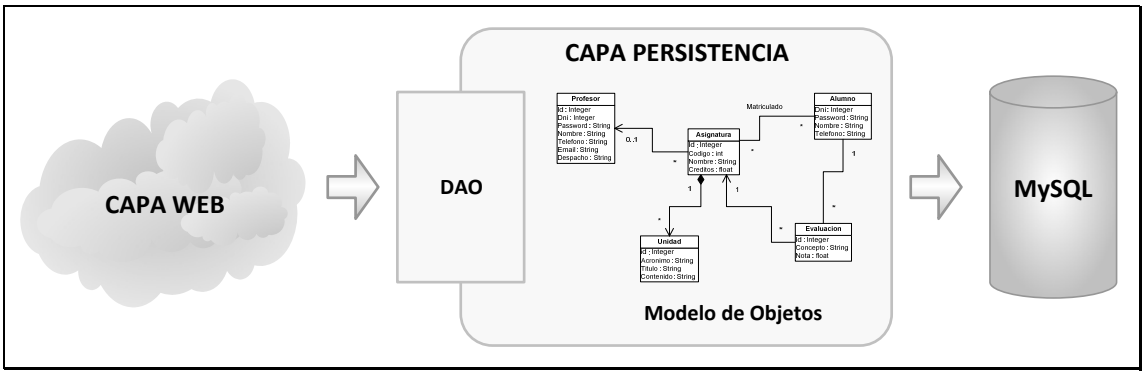

Figura 1: Arquitectura de la capa de persistencia

1.2. Modelo de objetos

Lo conforman el conjunto de clases (en este punto nos centramos únicamente en las entidades persistentes) y sus enlaces. La Figura 2 muestra el diagrama de clases con dichas entidades y enlaces.

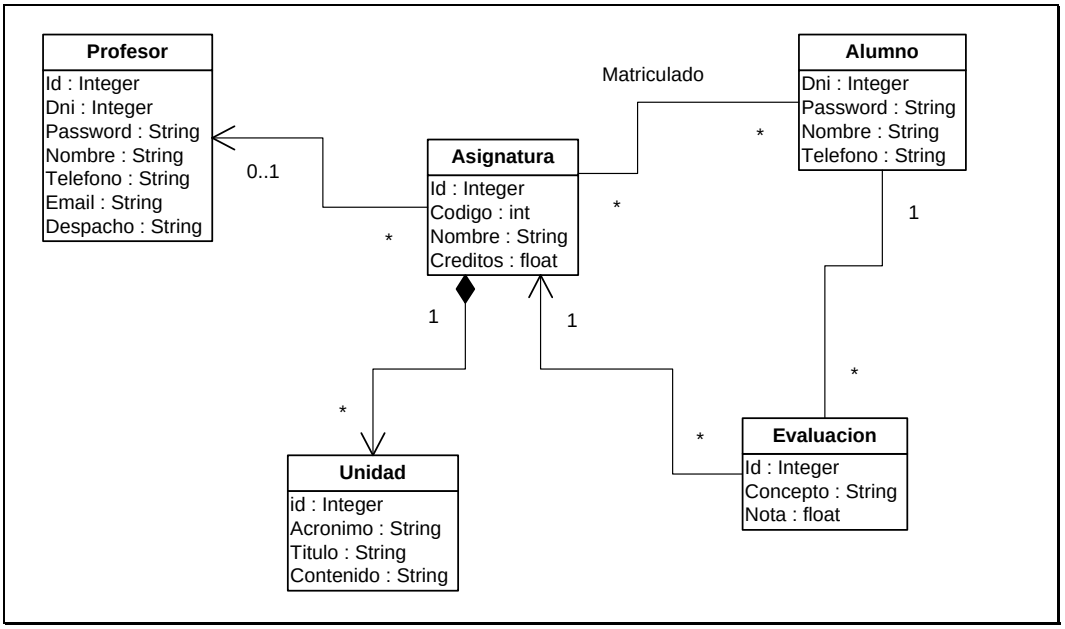

Figura 2: Diagrama de clases (persistentes)

Nota: en todos los casos el identificador o clave primaria (en la base de datos) de todas las clases será el atributo que aparece en primer lugar.

1.2.1 Especificación de restricciones

A continuación se presentan algunas restricciones a cumplir en el mapeo de las clases persistentes en la base de datos y que por falta de expresividad del lenguaje UML en algunos casos y por simplicidad en otros no se han mostrado en el diagrama de clases.

Profesor

- El identificador será asignado por la base de datos automáticamente.
- El nombre del profesor debe ser único y requerido.

Asignatura

- El identificador será asignado por la base de datos automáticamente.
- El código de la asignatura y el nombre son requeridos. Además el código debe ser único.
- Debe aplicarse carga perezosa y tratamiento en cascada entre las asignaturas y sus unidades (en el sentido de la relación estrictamente indicado).
- Todas las unidades deben estar vinculadas exactamente a una asignatura, es decir, no puede haber unidades sin asignaturas.
- Debe aplicarse carga perezosa entre las asignaturas y los alumnos matriculados (en el sentido de la relación estrictamente indicado).

Unidad

- El identificador será asignado por la base de datos automáticamente.
- El acrónimo de la unidad y el título son requeridos. Además el acrónimo debe ser único.

Alumno

- El identificador (DNI) será asignado por el usuario (programa).
- El nombre de la asignatura es requerido.
- Debe aplicarse carga perezosa y tratamiento en cascada entre los alumnos y sus evaluaciones (en el sentido de la relación estrictamente indicado).
- Debe aplicarse carga perezosa entre los alumnos y las asignaturas en las que están matriculados (en el sentido de la relación estrictamente indicado).

Evaluación

- El identificador será asignado por la base de datos automáticamente.
- El concepto y la nota son requeridas.
- Todas las evaluaciones deben estar vinculadas exactamente a una asignatura y a un alumno, es decir, no puede haber evaluaciones sin asignatura o alumno.

1.2.2 Métodos

Todas las clases, por exigencias de Hibernate, deben poseer la correspondiente pareja de métodos get/set para sus atributos, así como un constructor sin argumentos.

Adicionalmente, puede ser útil en el futuro (dependiendo de cómo se implemente la capa DAO) incluir el siguiente método de negocio en la clase **Asignatura**:

- *boolean estaMatriculado(Alumno)*

  Comprueba si un alumno se encuentra entre la lista de alumnos matriculados de la asignatura.

1.2.3 Consideraciones sobre la implementación

Implementa las clases de esta capa en un paquete llamado **iso3.pt.model**.

1.3. Capa DAO

Esta capa será la encargada de manejar el modelo de objetos persistentes descrito y de ofrecer a la capa web una interfaz con las operaciones necesarias para cumplir con la funcionalidad que a ésta se le requiere.

1.3.1 Interfaz

Seguidamente se listan los métodos que la capa DAO expone a modo de interfaz. En todos los casos el nombre del método es perfectamente descriptivo, razón por la cual no se requieren más explicaciones.

```
public Profesor getProfesor(int idAsignatura);
public Set<Alumno> getAlumnos(int idAsignatura);
public List<Evaluacion> getEvaluacionesOrderedByAsignatura(int idAlumno);
public Set<Evaluacion> getEvaluaciones(int idAsignatura, int idAlumno);
public void addEvaluacion(String concepto, float nota, int idAsignatura, int idAlumno);
public Set<Unidad> getUnidades(int idAsignatura);
public Set<Asignatura> getAsignaturas();
public Alumno getAlumno(int id);
public Asignatura getAsignatura(int id);
public Alumno loginAlumno(int dni, String pass) throws UserNotFoundException, IncorrectPasswordException;
public Set<Asignatura> getAsignaturas(int idAlumno);
public void matricular(int idAlumno, int idAsignatura);
public void desmatricular(int idAlumno, int idAsignatura);
public Profesor loginProfesor(int dni, String pass) throws UserNotFoundException, IncorrectPasswordException;
public Set<Asignatura> getAsignaturasProfesor(int idProfesor);
public Profesor getProfesorByDni(int dni) throws UserNotFoundException;
public List<Evaluacion> getEvaluacionesAsignatura(int idAsignatura);
```

1.3.2 Consideraciones sobre la implementación

Algunas buenas prácticas de codificación para la clase que implemente la capa DAO:

- Que siga el patrón Singleton.
- Dado que muchas de las operaciones de negocio giran en torno a las asignaturas, es conveniente que esta capa mantenga en caché e indexadas por su identificador toda la lista de asignaturas de forma que su búsqueda sea lo más rápida posible y evite el acceso a la base de datos.
- Esta capa deberá implementarse en un paquete llamado **iso3.pt.dao**. A la clase principal puedes llamarla por ejemplo **PtDAO**.

1.4. Otras consideraciones para la realización de la práctica

Es muy conveniente que vayas testeando poco a poco el trabajo que vayas realizando. La mejor forma de hacerlo es colocar un método *main()* en la clase que implemente la capa DAO e ir testeando los distintos objetos del modelo de objetos y posteriormente los diferentes métodos de la interfaz de la capa DAO.

El resultado final de esta fase debiera ser un método *main()* que pruebe todos los métodos de la interfaz de la capa DAO. Obviamente, cuando se incorpore la capa web este método dejará de tener utilidad.