Enunciado del Trabajo de JPA - Multas

Se parte del siguiente conjunto de tablas que modelan la gestión de incidencias de unos conductores que viven en grupos y comparten vehiculo por domicilios (se proporciona el script SQL, con TODAS las restricciones adicionales y una carga de datos inicial, recomendándose su revisión con detalle):

- ***Vehiculo(idauto (PK), nombre, direccion, cp, ciudad)***
- ***Conductor(nif (PK), nombre, apellido, direccion, cp, ciudad, puntos, idauto (FK -> Vehiculo))***
- ***TipoIncidencia(id (PK), descripcion, valor)***
- ***Incidencia(fecha (PK), nif (PK, FK -> Conductor), anotacion, idtipo (FK -> TipoIncidencia))***

- Donde los campos con PK forman la clave primaria de la tabla correspondiente.
- En la tabla `Conductor`, el campo `idauto` es clave foránea de `Vehiculo`.
- En la tabla `Incidencia`, el campo `nif` es clave foránea de `Conductor` e `idtipo` es clave foránea de `TipoIndicencia`.

Sobre dicho modelo de tablas, se pide implementar las siguientes transacciones, teniendo en cuenta los comentarios adicionales:

1- `public void insertarIncidencia(Date fecha, String nif, long tipo) throws PersistenceException;`
   - Inserta una nueva incidencia asociada a un conductor y tipo, descontando los puntos correspondientes a los puntos actuales del conductor.
   - Si el conductor no tiene puntos suficientes a descontar (no puede quedar en negativos) se genera una excepción.

2- `public void indultar(String nif) throws PersistenceException;`
   - Indulta a un conductor, borrando todas las incidencias que ha cometido y restaurando sus puntos al máximo (12 puntos).

3- `public List<Vehiculo> consultarVehiculos() throws PersistenceException;`
   - Consulta todos los vehículos. En este caso en particular es importante recuperar **toda la información vinculada a los vehículos, conductores e incidencias con su tipo,** para su visualización posterior.
   - Para su resolución es **obligatorio utilizar un grafo de entidades** que recupere toda la información solicitada (vehiculos, conductores e incidencias).

Nota: se ha detectado algún problema con el uso de `List` en *Hibernate* a la hora de cargas de grafos. Como alternativa, se sugiere cambiar en el modelo de entidades al tipo `Set` para las colecciones a varios, en lugar de `List`. En todo caso el tipo de datos retornado por la transacción siempre será `List` con independencia del uso de `Set` en el modelo de clases

persistentes.

Los códigos de las excepciones y codificaciones de error, se proporcionan, en `IncidentException`[1] e `IncidentError` respectivamente, en el paquete `es.ubu.lsi.service.multas`.

- Queda a juicio del alumnado generar las correspondientes excepciones con su correspondiente código de error en cada transacción.
- Si se considera que falta algún error en la enumeración, se pueden añadir a los valores del tipo enumerado. No se pueden eliminar los valores ya definidos.

Se proporciona una **batería de pruebas mínima** con tests de ejemplo sobre las transacciones solicitadas. El conjunto de pruebas garantiza un mínimo funcionamiento, pero deben tenerse en cuenta e implementarse todos los posibles casos de error adicionales que se puedan dar en las transacciones (datos de entrada incorrectos, lógica de negocio incorrecta, etc.)

Resolución del Trabajo

Para la realización del trabajo se seguirá el mismo diseño ya utilizado en las **dos últimas sesiones prácticas previas.** Por lo tanto, se seguirá utilizando el patrón de diseño DAO junto con JPA.

Se proporciona el esqueleto del proyecto JPA a completar en UBUVirtual.

En `es.ubu.lsi.service` está la **interfaz de servicio** `Service` que debe ser implementada, conteniendo los métodos correspondientes a las transacciones solicitadas.

Se asume que se utilizan las bibliotecas de usuario de sesiones anteriores de prácticas con JPA (proyecto `user_lib_JPA` y bibliotecas de usuario asociadas). En concreto son necesarias, **cuatro bibliotecas de usuario: FileSystemContext, Hibernate, Oracle** y **SLF4J**.

Se proporciona el script SQL en el subdirectorio `./sql/` del proyecto con nombre `script.sql`. Dicho script **NO debe ser modificado** puesto que se utiliza en la ejecución de las pruebas automáticas.

Código a implementar y orden sugerido de implementación

1- Paquete `es.ubu.lsi.model.multas` con las clases/entidades persistentes de JPA necesarias para el modelo de datos.

- Dado que existen atributos repetidos en dos tablas (i.e., direccion, cp, ciudad) se pide implementar esto con un tipo embebido adicional, de nombre DireccionPostal y que puede ser reutilizado tanto en Vehiculo como en Conductor.
- Nota: para evitar errores con el uso de grafos de entidades y la ejecución de los tests, se recomienda colecciones de tipo Set en lugar de tipo List para las relaciones a varios.

2- Paquete `es.ubu.lsi.dao.multas`: con los DAO necesarios que heredan de `JpaDAO`.

- Se recuerda que en todos estos DAO se debe implementar el método abstracto `findAll` con la típica consulta que devuelve TODAS las instancias de la entidad correspondiente.

3- Paquete `es.ubu.lsi.service.multas`: implementación del servicio concreto

[1]Se han añadido más constructores que los utilizados en ejercicios anteriores por si fueran necesarios.

`ServiceImpl` (ojo: **`extends`** `PersistenceService` **`implements`** `Service`) dando código concreto al cuerpo de los métodos (transacciones) de la interface `Service`.

4- Paquete `es.ubu.lsi.service.test`: implementación de la clase `TestClient`, con los tests automáticos, sobre la implementación de las transacciones con JPA. Se proporciona ya el código de esta clase.

5- En caso de que quieran ampliarse con tests adicionales, se copiará el código original y se ampliará en una clase `TestClientAlumno` en ese mismo paquete.

Pruebas "automáticas"

Las pruebas "automáticas" adicionales se programarán en la clase `es.ubu.lsi.test.TestClient`. Dicha clase contiene un método `main` para ejecutar los tests y unos tests básicos sobre las transacciones implementadas.

Esas pruebas están implementadas utilizando el servicio JPA y en algún caso se puede combinar con código JDBC (utilizando un pool de conexiones con JNDI), tal y como se ha venido trabajando en la asignatura previamente en la parte de JDBC.

Si se implementa algún test adicional (en otra clase `TestClientAlumno`), en que los datos de entrada pudieran generar **varios errores**, se deja a decisión del alumnado cuál de los errores se recoge y lanza en primer lugar, no prefijando ningún orden particular.

La salida esperada con los tests ya proporcionados en `TestClient`, debería ser similar a:

```
Iniciando...
Probando el servicio...
Framework y servicio iniciado...
Insertar incidencia correcta
          OK incidencia bien insertada
          OK actualiza bien los puntos del conductor
Insertar incidencia con tipo erróneo
          OK detecta correctamente que NO existe ese tipo de incidencia
Indulto del conductor...
          OK todas las incidencias borradas del conductor indultado
          OK puntos bien iniciados con indulto
          OK el número de incidencias de otros conductores es correcto
Indultar a un conductor que no existe
          OK detecta correctamente que NO existe ese conductor
Información completa con grafos de entidades...
Vehiculo [idauto=ABC, nombre=FORD, direccionPostal=DireccionPostal [direccion=Avda. Palencia 45, codigoPostal=09001, ciudad=Burgos]]
          Conductor [nif=10000000A, nombre=Juana, apellido=Manzanal, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=3]
                    Incidencia [id=IncidenciaPK [fecha=2019-05-15 16:00:00.0, nif=10000000A], anotacion=null, conductor=Conductor [nif=10000000A, nombre=Juana, apellido=Manzanal, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=3], tipoIncidencia=TipoIncidencia [id=3, descripcion=Moderada, valor=3]]
                    Incidencia [id=IncidenciaPK [fecha=2019-04-11 12:00:00.0, nif=10000000A], anotacion=Exceso de velocidad en calzada de pueblo, conductor=Conductor [nif=10000000A, nombre=Juana, apellido=Manzanal, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=3], tipoIncidencia=TipoIncidencia [id=2, descripcion=Grave, valor=6]]
          Conductor [nif=10000000B, nombre=Javier, apellido=Calle, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, codigoPostal=09003, ciudad=Burgos], puntos=6]
                    Incidencia [id=IncidenciaPK [fecha=2019-04-12 11:00:00.0, nif=10000000B], anotacion=Falta grave con semáforo, conductor=Conductor [nif=10000000B, nombre=Javier, apellido=Calle, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, codigoPostal=09003, ciudad=Burgos], puntos=6], tipoIncidencia=TipoIncidencia [id=2, descripcion=Grave, valor=6]]
          Conductor [nif=10000000C, nombre=Jimena, apellido=Plaza, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=12]
Vehiculo [idauto=MNP, nombre=CITROEN, direccionPostal=DireccionPostal [direccion=C/Progreso 10, codigoPostal=09001, ciudad=Burgos]]
          Conductor [nif=30000000B, nombre=Rosa, apellido=Manzanedo, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, codigoPostal=09003, ciudad=Burgos], puntos=6]
                    Incidencia [id=IncidenciaPK [fecha=2019-04-13 15:00:00.0, nif=30000000B], anotacion=Falta grave, conductor=Conductor [nif=30000000B, nombre=Rosa, apellido=Manzanedo, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, codigoPostal=09003, ciudad=Burgos], puntos=6], tipoIncidencia=TipoIncidencia [id=2, descripcion=Grave, valor=6]]
          Conductor [nif=30000000C, nombre=Roberto, apellido=Manzanita, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=0]
                    Incidencia [id=IncidenciaPK [fecha=2019-04-13 16:00:00.0, nif=30000000C], anotacion=Falta muy grave, conductor=Conductor [nif=30000000C, nombre=Roberto, apellido=Manzanita, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=0], tipoIncidencia=TipoIncidencia [id=1, descripcion=Muy grave, valor=12]]
          Conductor [nif=30000000A, nombre=Raquel, apellido=Del Barrio, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=9]
                    Incidencia [id=IncidenciaPK [fecha=2019-04-13 14:00:00.0, nif=30000000A], anotacion=Falta moderada, conductor=Conductor [nif=30000000A, nombre=Raquel, apellido=Del Barrio, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=9], tipoIncidencia=TipoIncidencia [id=3, descripcion=Moderada, valor=3]]
```

```
Vehiculo [idauto=XYZ, nombre=MERCEDES, direccionPostal=DireccionPostal [direccion=C/Obdulio 2, codigoPostal=09001, ciudad=Burgos]]
        Conductor [nif=20000000B, nombre=Pedro, apellido=Medina, direccionPostal=DireccionPostal [direccion=C/Vitoria 57, codigoPostal=09003, ciudad=Burgos], puntos=12]
        Conductor [nif=20000000A, nombre=Paloma, apellido=Del Barrio, direccionPostal=DireccionPostal [direccion=C/Vitoria 56, codigoPostal=09003, ciudad=Burgos], puntos=12]
        Conductor [nif=20000000C, nombre=Pablo, apellido=Torquemada, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=3]
                Incidencia [id=IncidenciaPK [fecha=2019-04-12 12:00:00.0, nif=20000000C], anotacion=Falta grave con semáforo, conductor=Conductor [nif=20000000C, nombre=Pablo, apellido=Torquemada, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=3], tipoIncidencia=TipoIncidencia [id=2, descripcion=Grave, valor=6]]
                Incidencia [id=IncidenciaPK [fecha=2019-04-12 13:00:00.0, nif=20000000C], anotacion=Falta grave posterior con semáforo, conductor=Conductor [nif=20000000C, nombre=Pablo, apellido=Torquemada, direccionPostal=DireccionPostal [direccion=C/Vitoria 58, codigoPostal=09003, ciudad=Burgos], puntos=3], tipoIncidencia=TipoIncidencia [id=3, descripcion=Moderada, valor=3]]
OK Sin excepciones en la consulta completa y acceso posterior
FIN.............
```

El resultado en el último test, sobre la consulta completa de multas, puede variar dado que depende de cómo se redefina el método `toString` de las correspondientes clases del modelo.

Condiciones de realización

- La práctica se realizará segun los grupos ya establecidos de otras practicas
- Para aclarar cualquier duda utilizar el foro habilitado. En caso de no aclararse las dudas en el foro, dirigirse al profesor en horario de tutorías.
- La realización de la prueba está sujeta al reglamento de evaluación de la UBU, aplicándose en caso de plagio.
- En caso de detectarse alguna irregularidad podría solicitarse una defensa oral de la misma en horario de tutorías a fijar con los responsables de la asignatura.

Comentarios adicionales

Además es **muy importante** tomar en consideración además los siguientes comentarios para la resolución del proyecto:

- La unidad de persistencia JPA debe llamarse "**`Multas`**".
- En este ejercicio los DAO deben incluir métodos adicionales. Es decisión del alumnado qué clases DAO son necesarias y qué código colocar en cada uno.
- Se considera que TODAS las asociaciones en el modelo de clases son bidireccionales.
- **En UBUVirtual se proporciona el proyecto con la solución parcial a completar por los alumnos. Se recuerda que debe ajustarse correctamente el `classpath` con las bibliotecas de usuario necesarias.**

Condiciones de entrega

- Se enviará un fichero .zip a través de la plataforma completando la tarea **[ABD] Entrega de Trabajo JPA 2022 1C.**
- Dicho fichero `.zip` contendrá TODOS los ficheros necesarios (fuentes, recursos, configuración de log, etc.) y utilizar rutas relativas, para poder ejecutarse sin problemas en

otro ordenador, con un Oracle Express 11g R2 que tenga un esquema para el usuario `HR` con constraseña `hr`. Los ficheros fuente estarán codificados en UTF-8.

- Se asumirá que en el equipo donde se corrige, se dispone de un proyecto de Eclipse (o equivalente) con nombre **user_lib_JPA** con las bibliotecas de usuario ya definidas y utilizadas en las sesiones de prácticas anteriores: **FileSystemContext**, **Hibernate**, **Oracle** y **SLF4J** Es obligatorio respetar el nombre y estructura de estas bibliotecas de usuario y **NO es necesario entregarlas.**

- El fichero comprimido seguirá el siguiente formato de nombre, **sin utilizar tildes**:
  - **`Nombre PrimerApellido-Nombre.zip`**
    Ej: si el alumno es Pablo García su fichero será **`Garcia-Pablo.zip`** si hay varios participantes se concatenaran los nombres
  - **Aunque la práctica se haga por grupos, se enviará por parte de cada uno de los dos integrantes.**

- **NO se admiten envíos posteriores a la fecha y hora límite, ni a través de otro medio que no sea la entrega de la tarea en UBUVirtual. Si no se respetan las normas de envío la calificación es cero.**

Criterios de valoración.

En caso de no compilar, la calificación, será de cero en este trabajo. Para la corrección y valoración también se tendrá en cuenta los resultados de la ejecución de los tests proporcionados.

A continuación se enumeran los criterios y pesos en la calificación de la práctica:

- **Modelo de entidades persistentes (15%)** (sin contar consultas, ni grafo de entidades)
- **Clases DAO (10%)**
- **Consultas y grafos de entidades (en modelo y DAOs) (25%)**
- **Correcta ejecución de las transacciones (50%)**

En la corrección **se penalizarán mucho los errores graves,** en particular los ya señalados en UBUVirtual en el documento **FAQ - Errores frecuentes con JPA**.

El trabajo tiene una **valoración final del 20% de la calificación final (2 puntos sobre 10),**