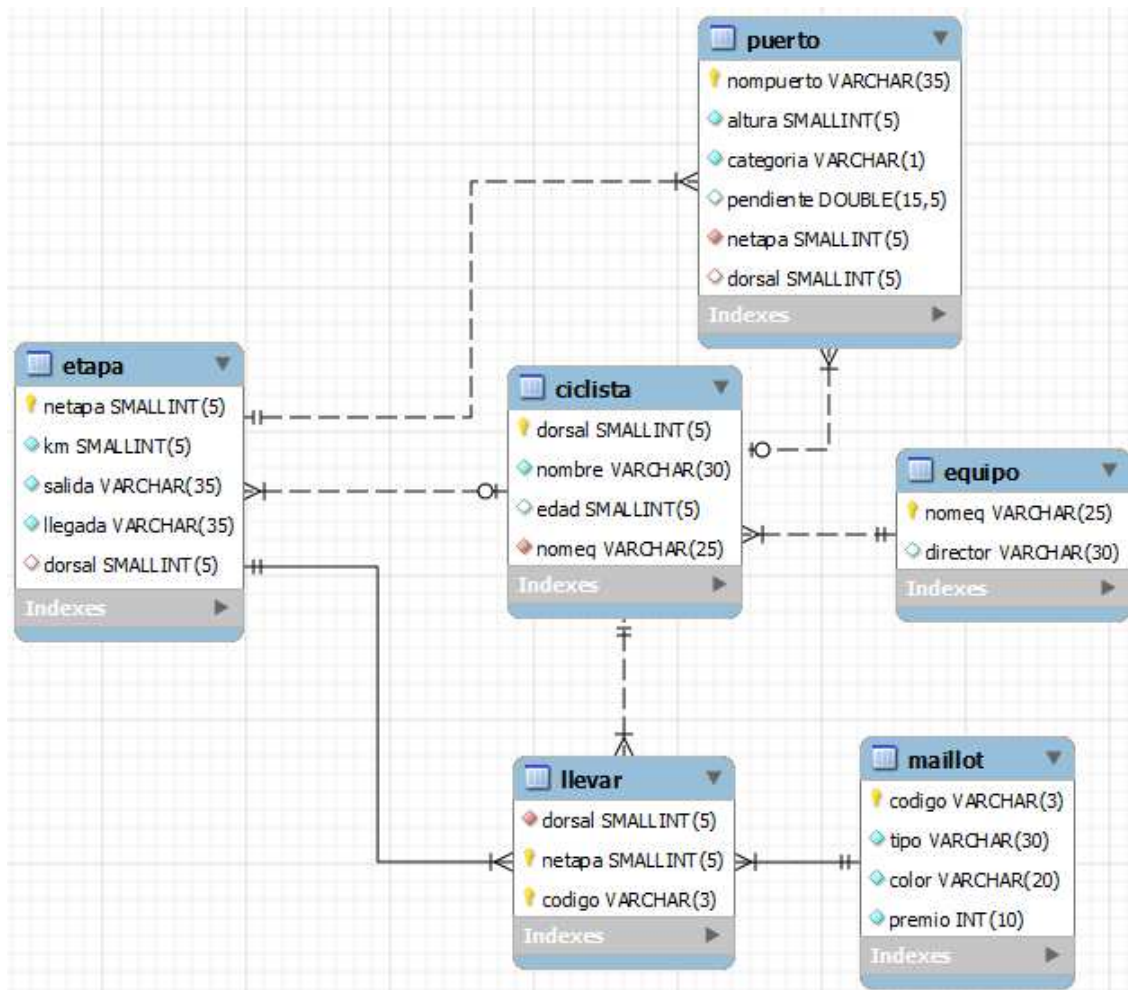


1. Dado el Diagrama E-R de la base de datos Ciclismo



a) Crea el proyecto ad.04ORM.01ciclismo y configúralo para que pueda utilizar Hibernate:

- Añadirle el proyecto utiles.
- Añadirle la user library HibernateLib
- Añadir un fichero de configuración hibernate.cfg.xml
- Añadir el fichero de configuración log4j.properties
- Añade un paquete pojos.
- Añade un paquete interfaz.

b) Crea los pojo Ciclista y Equipo.

- Crea los pojo Ciclista y Equipo, con constructores vacíos, métodos getter y setter, equals y hashCode. Utiliza convenientemente las anotaciones @Entity y @id para que la clase sea mapeada con Hibernate.
- En el pojo Ciclista añade la relación “muchos a uno” de Ciclista a Equipo. En la clase Ciclista el atributo que referencia al equipo se llamará “equipo”.
- En el pojo Equipo añade la relación “uno a muchos” de Equipo a Ciclista. En la clase Equipo el atributo que referencia a los ciclistas será un Set<Ciclista> y se llamará “ciclistas”.

- Añade al paquete interfaz un programa (clase `_01b01DirectorDelCiclista`) que, dado el dorsal de un ciclista introducido por el usuario, muestre el nombre del director de dicho ciclista.
 - Añade al paquete interfaz un programa (clase `_01b02CiclistasDeEquipo`) que, dado un nombre de equipo introducido por el usuario, muestre el nombre los ciclistas de dicho equipo.
- c) Crea el pojo Etapa (La tabla etapa tiene una relación uno a muchos con la tabla ciclista de manera que es posible saber qué ciclista ha ganado cada etapa).
- Crea el pojo Etapa, con constructores vacíos, métodos getter y setter, equals y hashCode. Utiliza convenientemente las anotaciones `@Entity` y `@id` para que la clase sea mapeada con Hibernate.
 - Añade al pojo Etapa la relación “muchos a uno” de Etapa a Ciclista. En el pojo Etapa el atributo que referencia al ganador de la etapa se llamará “ganador”.
 - En el pojo Ciclista añade la relación “uno a muchos” de Ciclista a Etapa. En la clase Ciclista el atributo que referencia a las etapas ganadas se llamará “etapasGanadas” y será un `Set<Etapa>`
 - Añade al paquete interfaz un programa (clase `_01c01GanadorDeEtapa`) que, dado un número de etapa introducido por el usuario, muestre el nombre del ciclista ganador de dicha etapa.
 - Añade al paquete interfaz un programa (clase `_01c02EtapasGanadasPorCiclista`) que, dado el dorsal de un ciclista introducido por el usuario, muestre la ciudad de salida y de llegada de las etapas ganadas por dicho ciclista.
 - Añade al paquete interfaz un programa (clase `_01c03EtapasGanadasPorEquipo`) que, dado un nombre de equipo introducido por el usuario, muestre el nombre de sus ciclistas junto con el número de etapas que ha ganado cada uno de ellos.
- d) Crea el pojo Puerto (La tabla puerto tiene una relación uno a muchos con la tabla etapa de manera que es posible saber a qué etapa pertenece cada puerto. Por otra parte tiene una relación uno a muchos con la tabla Ciclista de manera que es posible saber que ciclista ha ganado el puerto).
- Crea el pojo Puerto, con constructores vacíos, métodos getter y setter, equals y hashCode. Utiliza convenientemente las anotaciones `@Entity` y `@id` para que la clase sea mapeada con Hibernate.
 - Añade al pojo Puerto la relación “muchos a uno” de Puerto a Ciclista. En el pojo Puerto el atributo que referencia al ganador del puerto se llamará “ganador”.
 - En el pojo Ciclista añade la relación “uno a muchos” de Ciclista a Puerto. En la clase Ciclista el atributo que referencia a los puertos ganados se llamará “puertosGanados” y será un `Set<Puerto>`
 - Añade al pojo Puerto la relación “muchos a uno” de Puerto a Etapa. En el pojo Puerto el atributo que referencia a la etapa se llamará “etapa”.

- En el pojo Etapa añade la relación “uno a muchos” de Etapa a Puerto. En la clase Etapa el atributo que referencia a los puertos que contiene se llamará “puertos” y será un Set<Puerto>
- Añade al paquete interfaz un programa (clase _01d01GanadorDePuerto) que, dado un nombre de puerto introducido por el usuario, muestre el nombre del ciclista ganador de dicho puerto.
- Añade al paquete interfaz un programa (clase _01d02PuertosGanadosPorCiclista) que, dado el dorsal de un ciclista introducido por el usuario, muestre el nombre, la altura y la categoría de los puertos ganados por dicho ciclista.
- Añade al paquete interfaz un programa (clase _01d03GanadoresDeEtapayPuerto) que muestre el nombre de todos los ciclistas que ganaron una etapa y al mismo tiempo algún puerto perteneciente a dicha etapa.

2. Añade al paquete interfaz los siguientes programas:

- (_02aCrearEquipo) Programa que solicite al usuario el nombre de un equipo y el nombre del director y añada un nuevo equipo con los datos indicados. ¿En caso del que el equipo ya exista, qué excepción se produce? Modifica el programa capturando la excepción para avisar al usuario en caso de que el equipo introducido ya exista.
- (_02bModificarEquipo) Programa que solicite al usuario el nombre de un equipo y el nombre del director y modifique el equipo indicado para que el director pase a ser el indicado. Si el equipo indicado no existe se avisará al usuario.
- (_02cCambiarDeEquipo) Programa que solicite al usuario el dorsal de un ciclista y el nombre de un equipo y cambie al ciclista al equipo indicado. El programa mostrará mensajes de error en caso de que el ciclista indicado no exista, en caso de que el equipo indicado no exista y en caso de que el equipo indicado sea el mismo al que pertenece ya el ciclista.
- (_02dCrearCiclista) Programa que solicite al usuario el dorsal, nombre y nacimiento de un ciclista y el nombre de un equipo. El programa creará un nuevo ciclista con los datos indicados. El ciclista pertenecerá al equipo indicado. Se mostrarán mensajes de error en caso de que el ciclista ya exista y en caso de que el equipo no exista.

3. Ejecuta el siguiente fragmento de código y explica los resultados del siguiente fragmento de programa en los casos que se indican.

```
Equipo e = new Equipo();
e.setNombre("equipo1");
e.setDirector("director1");

Ciclista c = new Ciclista();
c.setNombre("ciclista1");
c.setNacimiento(new SimpleDateFormat("dd-MM-yyyy").parse("10-01-1970"));
c.setEquipo(e);
e.getCiclistas().add(c);

sesion.save(c);
```

- No se ha especificado *cascade* en Ciclista ni en Equipo.

- b) Se ha especificado *cascade ALL* en Ciclista, pero no en Equipo
 - c) Se ha especificado *cascade ALL* en Equipo, pero no en Ciclista
4. Ejecuta el siguiente fragmento de código y explica los resultados del siguiente fragmento de programa en los casos que se indican.

```
Equipo e = new Equipo();
e.setNombre("equipo1");
e.setDirector("director1");

Ciclista c = new Ciclista();
c.setNombre("ciclista1");
c.setNacimiento(new SimpleDateFormat("dd-MM-yyyy").parse("10-01-1970"));
c.setEquipo(e);
e.getCiclistas().add(c);

sesion.save(e);
```

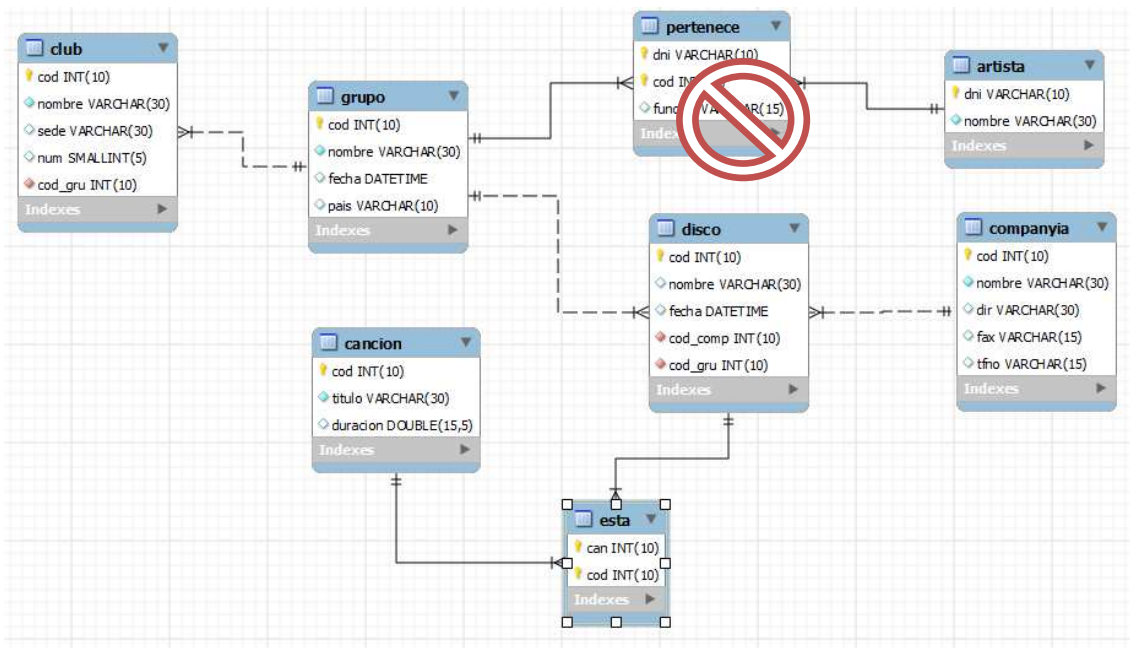
- a) No se ha especificado *cascade* en Ciclista ni en Equipo.
 - b) Se ha especificado *cascade ALL* en Ciclista, pero no en Equipo
 - c) Se ha especificado *cascade ALL* en Equipo, pero no en Ciclista
5. Modifica convenientemente el fragmento de código del ejercicio anterior para que funcione correctamente en caso de que no se haya definido actualizaciones en cascada.
6. Escribe un programa que elimine el equipo que indique el usuario y todos sus ciclistas. ¿Qué efecto tiene en este caso el atributo *cascade*?
7. Escribe un programa que elimine el equipo que indique el usuario pasando previamente sus ciclistas a otro equipo (que también indicará el usuario)
8. En el proyecto 03ORM.ciclismo crea una clase `_08HQLBasicas` para ejecutar las siguientes consultas HQL
- a) Ciclistas nacidos entre el 1/1/1979 y 31/12/1980.
 - b) Ciclistas cuyo nombre empiece o termine por A
 - c) Número total de ciclistas
 - d) Número total de ciclistas del equipo Banesto
 - e) Fecha de nacimiento del ciclista más joven
 - f) Tamaño medio de los nombres de los ciclistas
 - g) Ciclistas cuyo nombre tiene más de 15 caracteres.
 - h) Ciclistas cuyo nombre tiene más tamaño que la media.
 - i) Ciclistas ordenados por fecha de nacimiento de menor a mayor. A igual fecha, ordenados por nombre de mayor a menor.
9. En el proyecto 03ORM.ciclismo crea una clase `_09HQLVariasTablas` para ejecutar las siguientes consultas HQL. Ejecútalas previamente en el HQL Editor
- a) De cada etapa, ciudad de salida y llegada y nombre del ciclista que la ha ganado
 - b) De cada equipo, nombre, director y número de ciclistas que tiene.
 - c) De cada equipo, nombre del equipo y número de etapas que han ganado sus ciclistas.
 - d) De cada etapa, ciclista que la ha ganado y equipo al que pertenece.
 - e) Ciclistas que han ganado alguna etapa que salía o llegaba a Benidorm.

- f) Ciclistas que han ganado alguna etapa con salida y llegada de la misma ciudad
 - g) Equipos cuyos ciclistas han ganado alguna etapa con salida y llegada en la misma ciudad
 - h) De cada año, número de ciclistas han nacido ese año.
10. En el proyecto 03ORM.ciclismo crea una clase _10FiltrarCiclistas que muestre los ciclistas filtrándolos previamente. Para ello crea los siguientes métodos:
- a) List<Ciclista> getCiclistas(String nombreCiclista, String nombreEquipo, Date desdeNacimiento, Date hastaNacimiento, Integer etapasGanadas) donde:
 - i. nombreCiclista: Permitirá mostrar sólo los ciclistas cuyo nombre contenga el texto indicado. En caso de ser null no se tendrá en cuenta este filtro.
 - ii. nombreEquipo: Permitirá mostrar sólo los ciclistas pertenecientes al equipo indicado. En caso de ser null se mostrarán ciclistas de cualquier equipo.
 - iii. desdeNacimiento: Permitirá mostrar solo ciclistas cuya fecha de nacimiento sea igual o superior a la indicada. En caso de ser null no se tendrá en cuenta este filtro.
 - iv. hastaNacimiento: Permitirá mostrar solo ciclistas cuya fecha de nacimiento sea igual o inferior a la indicada. En caso de ser null no se tendrá en cuenta este filtro.
 - v. etapasGanadas: Permitirá mostrar sólo ciclistas que como mínimo hayan ganado el número de etapas indicado. En caso de ser null no se tendrá en cuenta las etapas ganadas.

Nota: El query tendrá que construirse dinámicamente a medida que se compruebe qué filtros hay que tener en cuenta y cuáles no, tal y como se hizo en el ejercicio 7 del tema de jdbc (repásalo)

- b) Utiliza el método main de la clase para probar el método con distintos filtros y mostrar los resultados por pantalla:

11. Dado el Diagrama ER de la base de datos MusicaConClavesAjenas (cuyo script encontrarás en el aula virtual)



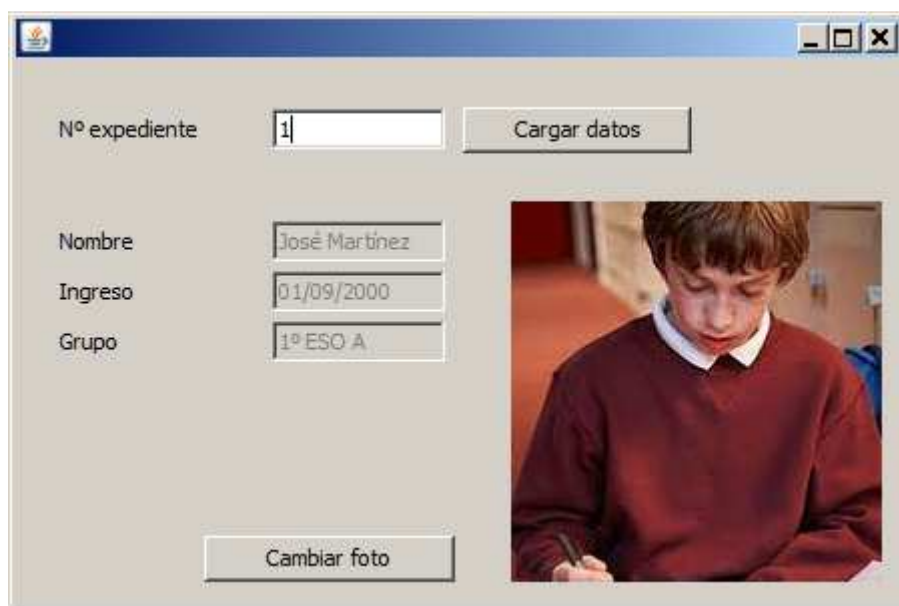
- Crea el proyecto ad.03ORM.musica y configúralo para que pueda utilizar Hibernate.
 - Crea los pojo's Artista, Club, Grupo, Compania, Disco y Cancion así como las relaciones uno a muchos y muchos a uno existentes.
 - Crea también la relación muchos a muchos entre Disco y Canción que determina en qué discos se encuentra una canción y qué canciones contiene cada disco. Ignora, sin embargo, la relación muchos a muchos que indica a que grupos pertenece un artista y viceversa.
 - Realiza pequeños programas que consideres convenientes para comprobar el correcto funcionamiento del trabajo realizado.
12. Crea un proyecto ad.03ORM.musica2 y genera los pojo y anotaciones correspondientes a la base de datos música utilizando para ello las HibernateTools. A continuación:
- Realiza las siguientes sentencias HQL:
 - De las canciones que duran más de 3 minutos, mostrar el título, la duración, el nombre del disco en que se encuentran y la fecha en la que se lanzó el disco.
 - En la consulta anterior hacer que solo se muestren canciones que están en discos que se lanzaron a partir de 1982 (puedes probar con otras fechas).
 - De cada grupo, el nombre del grupo, nombre de los artistas que lo integran y función que desempeña el artista dentro del grupo.
 - De cada artista que actúa como batería, grupo con el que lo hace y clubes en los que toca.
 - De cada compañía, nombre de la compañía y nombre de los grupos que guardan alguna relación con ella.

- vi. De cada compañía, nombre de la compañía y nombre de los artistas que guardan alguna relación con ella.
- b) Realiza los siguientes programas: Realiza el diseño en capas, es decir un método con los parámetros necesarios en la capa DAO y un programa para probar el método correspondiente en la capa interfaz:
 - i. Programa que cree un disco y una canción. La nueva canción pertenecerá al nuevo disco. El disco pertenecerá a un grupo y a una compañía ya existente. Se mostrarán (en la capa interfaz) los mensajes de error correspondientes en los siguientes casos:
 - La compañía a la que pertenece el disco no existe.
 - El grupo al que pertenece el disco no existe.
 - ii. Programa que, dado un código de grupo y el dni de un artista cambie la función que desempeña el artista en el grupo por la de "saxofón". Se mostrarán (en la capa interfaz) los mensajes de error correspondientes en los siguientes casos:
 - El grupo indicado no existe
 - El artista indicado no existe.
 - No se ha podido realizar la modificación por algún otro motivo.
 - iii. Programa que intercambie los discos de dos compañías dados sus códigos, es decir, los discos de una de las compañías pasarán a pertenecer a la otra y viceversa. Se mostrarán mensajes de error si:
 - alguna de las compañías no existe.
 - iv. Programa que cree un nuevo disco: Dado un código de grupo creará un nuevo disco que será una recopilación de canciones del grupo:
 - El nombre del disco se llamará "recopilatorio".
 - Tendrá la fecha actual.
 - Pertenecerá a una compañía de código dado, que debe existir (en caso contrario se mostrará error)
 - El nuevo disco estará formado por otras canciones del grupo, en concreto, la canción de menor duración de cada uno de sus anteriores discos.
 - v. Programa que aumente en un minuto la duración de las canciones de un disco, dado el código del disco. Se mostrará por pantalla el número de minutos de duración que ha aumentado el disco en total.

13. Colegio

- a) Ejecuta el script colegio.sql que se te ha entregado. Observa las tablas generadas, sus campos y sus relaciones.
- b) Crea un nuevo proyecto y genera, utilizando las Hibernate Tools, los pojo's y los ficheros de mapeo correspondientes a las tablas de la base de datos.
- c) Observa:
 - i. ¿Con qué tipo/modalidad de generación de clave primaria se han generado las entidades? Infórmate: ¿Cuál es el funcionamiento de dicha modalidad de generación de claves? ¿Qué otras modalidades existen?
 - ii. ¿Qué constructores se han generado en las clases de entidad?

- d) Haz un programa que inserte en la base de datos un nuevo grupo. Se debe avisar del error si ya existe un grupo con el mismo código. Una vez insertado, mostrar el identificador del grupo creado.
 - e) Haz un programa que inserte un nuevo alumno y lo asigne a un grupo. Se debe avisar del error si el equipo indicado no existe o si ya existe un alumno con el nº de expediente indicado. Una vez insertado, mostrar el identificador del alumno creado
 - f) Modifica el programa anterior para tener en cuenta el nº máximo de alumnos por grupo. En la tabla *grupo* el campo *maxalumnos* indica cuántos alumnos puede haber en cada grupo. Si se intenta rebasar el máximo se mostrará un mensaje y el alumno no se agregará.
 - g) Realiza un programa para modificar los datos de un grupo. Se debe avisar del error si se modifica el código por uno que ya existe, o si el *maxalumnos* se establece a un valor incorrecto, es decir, si el grupo tiene en ese momento más alumnos que los indicados.
14. Colegio Foto: Realizar un formulario (puede ser similar al de la imagen) que ...
- a) ... que cargue los datos de un alumno conocido su nº de expediente. El formulario debe incluir la foto del alumno.
 - b) ... disponga de un botón para modificar la foto del alumno.



15. Extraescolares ...

La base de datos dispone de las siguientes tablas:

- **Actividades:** id, descripción y fecha de realización de actividades extraescolares.
 - **Participación:** Indica qué alumnos participan en qué actividades y si han efectuado o no el pago. Esta tabla sostiene la relación muchos a muchos entre alumnos y actividades
- a) Incorpora al proyecto los POJO'S y los ficheros de mapeo correspondientes a actividades y participación, observa y responde ...

- i. ¿Cómo se mapea la relación muchos a muchos entre actividades y alumnos?
- ii. ¿Qué entidades nuevas han aparecido? ¿Cuál es el identificador de la entidad Participación?
- iii. ¿En qué tabla de la BD se guardará la información de los objetos ParticipacionId?
- iv. ¿Qué diferencias observas (en la base de datos) entre la fecha de ingreso de un alumno y la fecha de realización de una actividad.? ¿En ambos casos el atributo de las clases es de tipo Date ¿cómo se traduce esa diferencia en los ficheros de mapeo o en las anotaciones?
- v. En la base de datos, la tabla participación tiene un campo de tipo bit (que admite 0 o 1) para indicar si un alumno ha pagado o no su participación en la actividad. ¿Cómo se ha mapeado dicho campo en Java? Realiza alguna prueba para comprobar si 0 corresponde con true o con false.

b) Realizar un formulario (que puede ser similar al de la figura) que permita incluir y quitar a alumnos de una actividad.

The screenshot shows a software window titled "Visita al Bioparc" with a standard Windows-style title bar. The interface includes a dropdown menu at the top left labeled "--Filtrar por grupo ---". Below this is a list of student names: Angel Aguado, Barbara García, Bartolomé Nuñez, Carla Rodrigo, Carlos Fernandez, David Gómez, and Enrique Andreu. To the right of this list is a button labeled "Añadir". Further right is a list of activities: Visita al Bioparc, Visita al Oceanográfico, Salida al teatro, and Visita Centro de cálculo. Below the student list is a table with three columns: "Actividad", "Alumno", and "Pagado". The table contains two rows: "Visita al Bioparc" with "Manuel Sanz" and "Si", and "Visita al Bioparc" with "Alba Martorell" and "No". To the right of the table is a button labeled "Eliminar de la actividad".

Filtra los alumnos por grupo

Añade a **Angel Aguado** a la actividad **Visita al Bioparc**

Actividades existentes

Muestra los alumnos inscritos en la Actividad seleccionada

Elimina a **Manuel Sanz** de **Visita a Bioparc**

Actividad	Alumno	Pagado
Visita al Bioparc	Manuel Sanz	Si
Visita al Bioparc	Alba Martorell	No