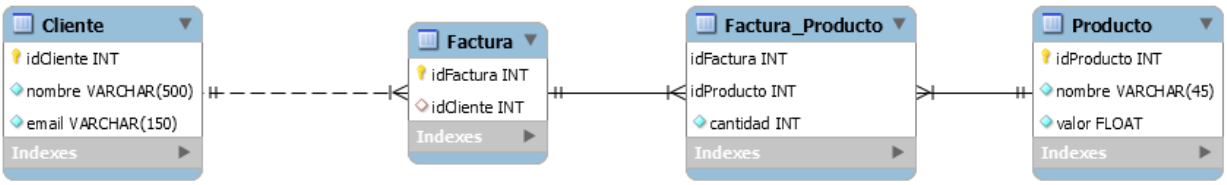
Trabajo Práctico No. 2: Persistencia

1) Introducción a JDBC
   a. Realizar la configuración de Derby con JDBC (Java) y testear la base de datos con algunas consultas sobre la tabla Persona.
      Video 1: https://www.loom.com/share/a9904b69ecc04892827b740b7d78432a
   b. Realizar la configuración de MySQL con JDBC (como sugerencia, puede utilizar también Docker) y testear la base de datos con las mismas consultas sobre la tabla Persona realizadas en el ejercicio 1.
      Video 2: https://www.loom.com/share/1014c5b740d5491690676ff448209959
   c. Refactorizar lo realizado en los resultados de a) y b) utilizando un patrón DAO que permita abstraer los detalles de la conexión a cada tipo de base de datos y las consultas. Crear una clase Persona para encapsular los datos de la misma, e implementar un servicio que retorne una lista de todas las personas cargadas en la base de datos.
   d. Enumerar brevemente las ventajas de la implementación basada en DAO del ejercicio anterior, respecto a las implementaciones de a) y b). Adicionalmente, considere posibles desventajas.

2) Introducción a JPA y Hibernate
   a. Re-implementar las consultas del ejercicio 1a, pero esta vez utilizando una implementación basada en JPA y Hibernate.
      Video 3: https://www.loom.com/share/5d807aa6d03349958f7be2a487ae51a6
   b. Re-implementar las consultas del ejercicio 1b, pero esta vez utilizando una implementación basada en JPA y Hibernate.
      Video 4: https://www.loom.com/share/fea19216d50f42aeb9ae2fc7e924ec4b
   c. Enumerar brevemente las ventajas de la implementación basada en JPA de a) y b), respecto a las implementaciones sin JPA del ejercicio 1. Adicionalmente, considere posibles desventajas.

3) Extender el diseño basado en DAO del ejercicio 1 para incorporar las implementaciones alternativas basadas en JPA y Hibernate del ejercicio 2. Para esto, puede utilizar patrones adicionales (por ej., Singleton, Factory). Crear una clase Persona para encapsular los datos de la misma, e implementar un servicio que retorne una lista de todas las personas cargadas en la base de datos.

4) Considere el siguiente diagrama de base de datos:

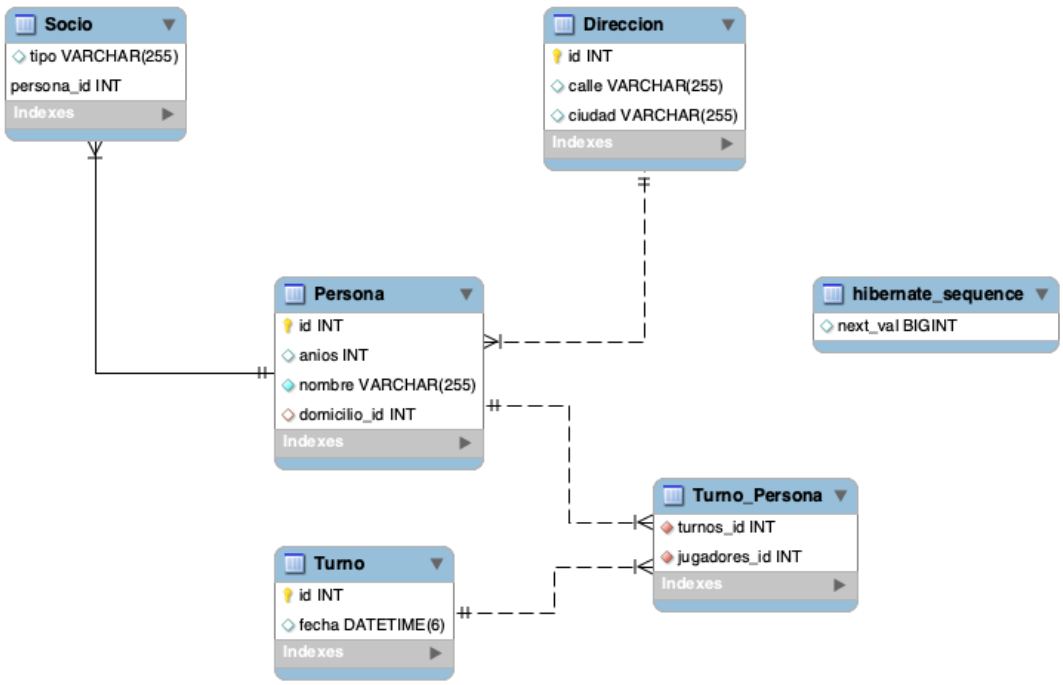


a. Crear un programa utilizando JDBC que cree el esquema de la base de datos.
b. Considerar los CSV dados y escribir un programa JDBC que cargue los datos a la base de datos. Considerar utilizar la biblioteca Apache Commons CSV, disponible en Maven central, para leer los archivos.

```
CSVParser parser = CSVFormat.DEFAULT.withHeader().parse(new
                   FileReader("productos.csv"));
for(CSVRecord row: parser) {
      System.out.println(row.get("idProducto"));
      System.out.println(row.get("nombre"));
      System.out.println(row.get("valor"));
}
```

c. Escribir un programa JDBC que retorne el producto que más recaudó. Se define “recaudación” como cantidad de productos vendidos multiplicado por su valor.
d. Escribir un programa JDBC que imprima una lista de clientes, ordenada por a cuál se le facturó más.

5) Implementar los mapeos de las entidades Turno, Dirección, Persona, y Socio a tablas en una base de datos relacional (Derby o MySQL), así como las relaciones entre ellos. Para testear, probar casos de altas, bajas y modificaciones de entidades.
Video 5: https://www.loom.com/share/5433281ad87342dcaafad8b787e0eeba



6) En base al ejercicio anterior, implementar las siguientes consultas:
   a. recuperar todas las personas asignadas a un turno.
   b. recuperar todas las personas asignadas a un turno, y marcar cuales de ellas son socios.
   c. recuperar todas las personas que viven en una ciudad predeterminada.
   d. En los casos anteriores, evaluar que sucede al utilizar las opciones FetchType.LAZY o FetchType.EAGER en las anotaciones. ¿Nota alguna diferencia?, ¿a qué se debe?

7) Suponga un sistema para gestionar torneos de futbol 7, donde las principales entidades son: Equipo, Jugador, y Torneo. Cada equipo incluye 7 jugadores (uno de ellos debe ser arquero), hasta 3 suplentes, y 1 director técnico. Cada equipo puede o no representar a una entidad o firma comercial. Los jugadores desempeñan una posición fija dentro del equipo (por ej., arquero, defensa, mediocampo, delantera). Un jugador solo puede estar fichado en un equipo. Un torneo tiene un nombre e incluye un conjunto prefijado de equipos.
   a. realizar un modelamiento orientado a objetos de las entidades y sus relaciones.
   b. diseñar un DER que permita mapear el modelo orientado a objetos anterior.
   c. implementar el mapeo entre a) y b) utilizando anotaciones de JPA.
   d. implementar servicios para altas, bajas y modificaciones de las distintas entidades
   e. implementar servicios para agregar jugadores a equipos, e inscribir equipos a torneos
   f. implementar servicios para buscar todos los jugadores de un equipo, y todos los jugadores de un torneo.

8) Extender el diseño anterior con la siguiente información. Al inicio del torneo se establecen los encuentros (todos contra todos). A medida que se van jugando los encuentros de cada fecha, se va llevado un registro del resultado de cada partido, de los goles que anotó cada

jugador, y de los puntos que obtuvo cada equipo en una partida que jugó. Luego de cada partido, puede que se reporten jugadores lesionados, o expulsados, que no podrán jugar partidos por un período de tiempo determinado. En caso que un equipo no pueda completar su formación, se declarará ganador al equipo contrario.

a. adecuar el modelo de objetos, y también el esquema de tablas.
b. agregar las consultas correspondientes para registrar la nueva información en los objetos y tablas del sistema.
c. implementar un servicio que genere la tabla de posiciones, a una fecha dada.
d. Implementar un servicio que genere la tabla de goleadores, a una fecha dada.

Nota: para testear los ejercicios se recomienda definir un conjunto de datos básico que contenga: nombres de jugadores y equipos.

9) **Ejercicio Integrador.** Considerar el diseño de un registro de estudiantes, con la siguiente información: nombres, apellido, edad, género, número de documento, ciudad de residencia, número de libreta universitaria, carrera(s) en la que está inscripto, antigüedad en cada una de esas carreras, y si se graduó o no.

a. Diseñar el diagrama de objetos y el diagrama DER correspondiente.
b. Implementar consultas para:

- dar de alta un estudiante
- matricular un estudiante en una carrera
- recuperar todos los estudiantes, y especificar algún criterio de ordenamiento simple.
- recuperar un estudiante, en base a su número de libreta universitaria.
- recuperar todos los estudiantes, en base a su género.
- recuperar las carreras con estudiantes inscriptos, y ordenar por cantidad de inscriptos.
- recuperar los estudiantes de una determinada carrera, filtrado por ciudad de residencia.

c. Generar un reporte de las carreras, que para cada carrera incluya información de los inscriptos y egresados por año. Se deben ordenar las carreras alfabéticamente, y presentar los años de manera cronológica.

Nota: las consultas deben ser resueltas mayormente en JPQL, y no en código Java.