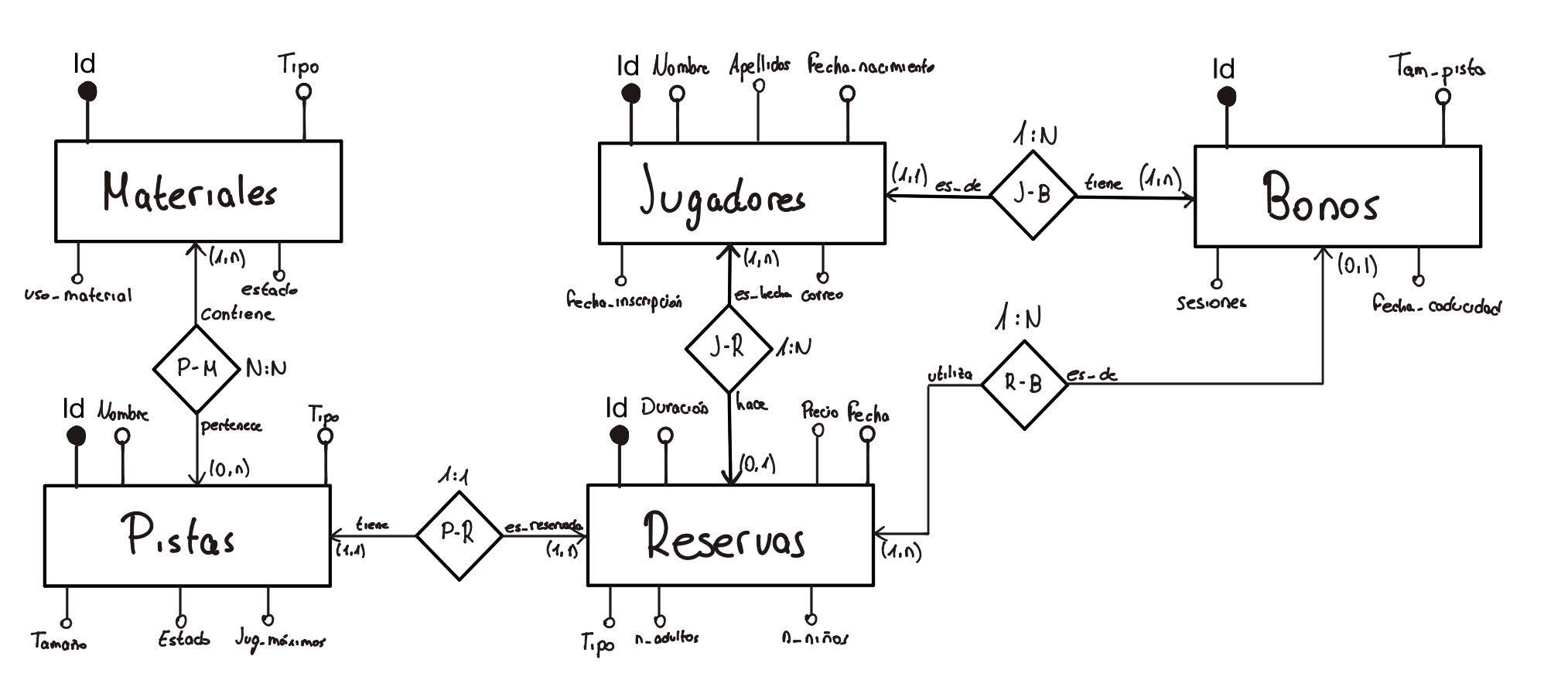
**Práctica 2**

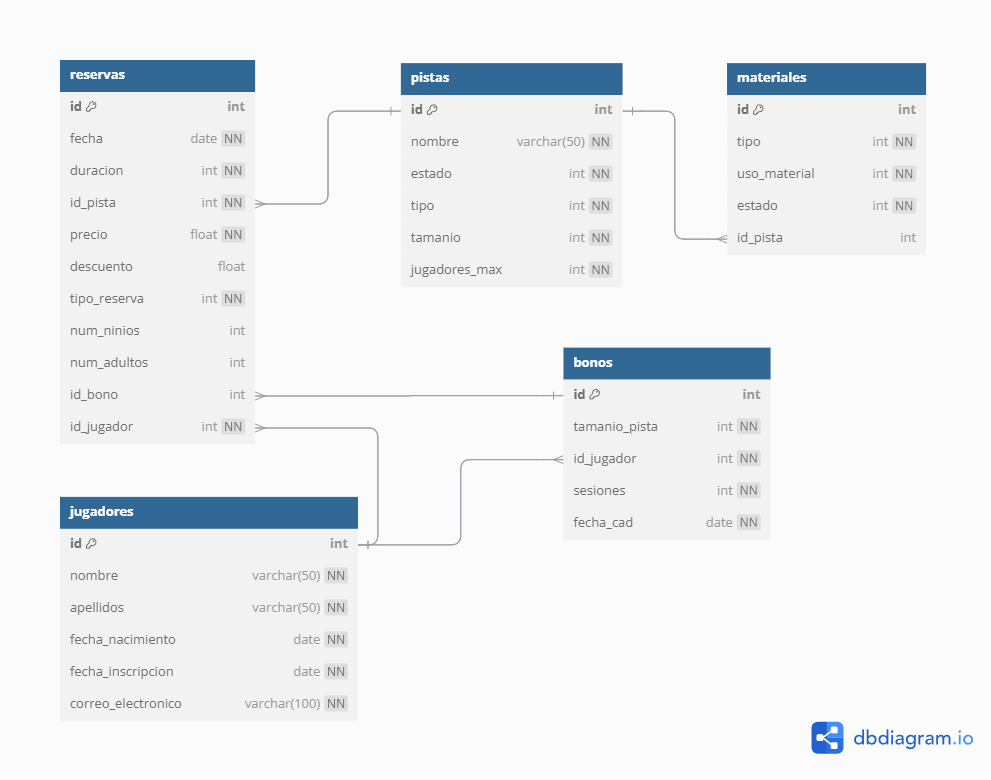
Para el desarrollo de la segunda práctica se han tomado ciertas decisiones de diseño:

* Eliminación del patrón factoría para la implementación de las reservas. En su lugar se ha optado por poner el campo tipo\_reserva para distinguir entre reservas infantiles, familiares y de adultos y una foreign key para hacer referencia al bono al que pertenece esa reserva (puede ser null en caso de que sea una reserva individual)
* Para las opciones de búsqueda de pistas, jugadores… se ha optado por usar el correo para los jugadores y el nombre para lo demás. El campo id es meramente identificativo y autoincrementable.
* Los atributos de tipo “enum” se han convertido en enteros para agilizar las consultas a la base de datos.

Ejercicio 1:

**Modelo Entidad-Interrelación:**



**Modelo relacional:**

Ejercicio 2:

#### **Estrategias Implementadas**

1. **Arquitectura de Programación en Tres Capas** Para estructurar y organizar el proyecto, adoptamos un enfoque basado en tres capas:
   * **Capa de Datos (data):**
     + Los objetos de acceso a datos (DAO) de las clases principales se encuentran en el paquete data.dao. Estos DAO encapsulan la lógica de interacción con la base de datos.
     + La configuración y gestión de la conexión a la base de datos están centralizadas en el paquete data.common.
   * **Capa de Negocio (business):**
     + Contiene los gestores y objetos de transferencia de datos (DTO) que implementan la lógica de negocio y sirven como puente entre las capas de datos y presentación.
   * **Capa de Presentación (display):**
     + Incluye los métodos main de las tres clases principales: Usuarios, Reservas y Pistas. Estos métodos manejan la interacción con el usuario final y actúan como punto de entrada para ejecutar el programa.
2. **Distribución del Trabajo** Dividimos el desarrollo del proyecto asignando las responsabilidades por clases principales, de manera que cada integrante del equipo se encargó de desarrollar y probar una clase en particular. Esta estrategia nos permitió avanzar en paralelo y minimizar conflictos en el código.
3. **Gestión Centralizada de Sentencias SQL y Configuración de Base de Datos**
   * Todas las sentencias SQL necesarias para las operaciones CRUD se almacenaron en el archivo sql.properties. Este archivo es cargado dinámicamente en el constructor de cada DAO, facilitando:
     + La centralización y mantenimiento de las consultas.
     + La posibilidad de realizar cambios sin modificar el código Java directamente.
   * Los parámetros de conexión a la base de datos (URL, usuario, contraseña, etc.) se encuentran en el archivo config.properties, lo que permite flexibilidad en caso de modificaciones en el entorno de despliegue.

#### 

#### **Problemas Encontrados**

1. **Inexperiencia del Equipo con Java** Una de las principales dificultades fue la falta de experiencia de varios integrantes en el lenguaje Java. Esto generó:
   * Retos en la comprensión de conceptos básicos como manejo de excepciones, clases y métodos, y estructura de paquetes.
   * Un incremento en el tiempo necesario para implementar y depurar el código.
2. **Implementación de la Programación por Capas** La adopción de una arquitectura en tres capas representó un desafío, especialmente en:
   * Entender cómo deben comunicarse las capas (display, business y data) sin romper la separación de responsabilidades.
   * Definir correctamente las funciones de los DTO, DAO y gestores.
3. **Fallas en la Base del Proyecto (Práctica 1)** No haber completado la Práctica 1 correctamente complicó el desarrollo de este proyecto, ya que:
   * No contábamos con una base sólida para la conexión y manejo de la base de datos.
   * Tuvimos que invertir tiempo en solucionar problemas básicos que ya deberían haber sido resueltos en dicha práctica.

**Webgrafía**

<https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>

[https://dbdiagram.io](https://dbdiagram.io/)

<https://www.youtube.com/watch?v=NjY-WA-jeJ8>

<https://chatgpt.com>