Backend de Aplicaciones

> ## Consigna – Pre-enunciado Turno 2
>
> **Contexto:** dataset *Board Games* (Kaggle)
>
> **Objetivo** Preparar una **aplicación Java (consola)** que deje lista **la base de datos H2 en memoria**, el **mapeo JPA/Hibernate**, y una **validación mínima**. Todo apuntado a que, en el día del parcial, solo tengas que agregar el **parseo del CSV** y los **procesos** que se soliciten.

---

1) Requisitos técnicos (obligatorios)

- **Java 17 o superior**, **Maven**.
- Librerías: **Lombok**, **JDBC**, **JPA/Hibernate**.
- **H2 en memoria** (embedded): obligatorio.
- **DDL y datos**: usar el archivo provisto `ddl_board_games.sql` (estructura y seed).
- **Convenciones de nombres**:
  - Tablas y columnas en **MAYÚSCULAS** (vienen así en el DDL).
  - Clases Java en **UpperCamelCase**; campos Java en **lowerCamelCase**.
  - Mapear nombres distintos con `@Column(name = "…")`.

---

2) Estructura de proyecto (SUGERIDA)

> **El alumno puede optar por la estructura que prefiera** y con la que se sienta más confiado para la realización del parcial. La que sigue es **solo una guía** alineada con los pasos que venimos usando en JDBC y JPA.

```
/ (root)
├─ src/
│  └─ main/
│     ├─ java/utnfc/isi/back/...        ← paquetes según criterio de
cátedra
│     │  ├─ infra/
│     │  │  ├─ DataSourceProvider.java             ← proveedor JDBC (H2
en memoria)
│     │  │  ├─ LocalEntityManagerProvider.java     ← proveedor de
EntityManager (JPA)
│     │  │  └─ DbInitializer.java                  ← ejecuta
ddl_board_games.sql vía JDBC
│     │  ├─ domain/                                ← entidades JPA
│     │  │  ├─ Category.java
│     │  │  ├─ Designer.java
│     │  │  ├─ Publisher.java
```

```
│     │  │  └─ BoardGame.java
│     │  └─ repo/                                   ← opcional (repos JPA
o EM directo)
│     ├─ resources/
│     │  ├─ META-INF/
│     │  │  └─ persistence.xml                      ← unidad de
persistencia JPA
│     │  └─ sql/
│     │     └─ ddl_board_games.sql                  ← DDL y seed provistos
│     └─ App.java                                   ← Main: orquesta init
+ validación
└─ pom.xml
```

---

3) Tareas a realizar

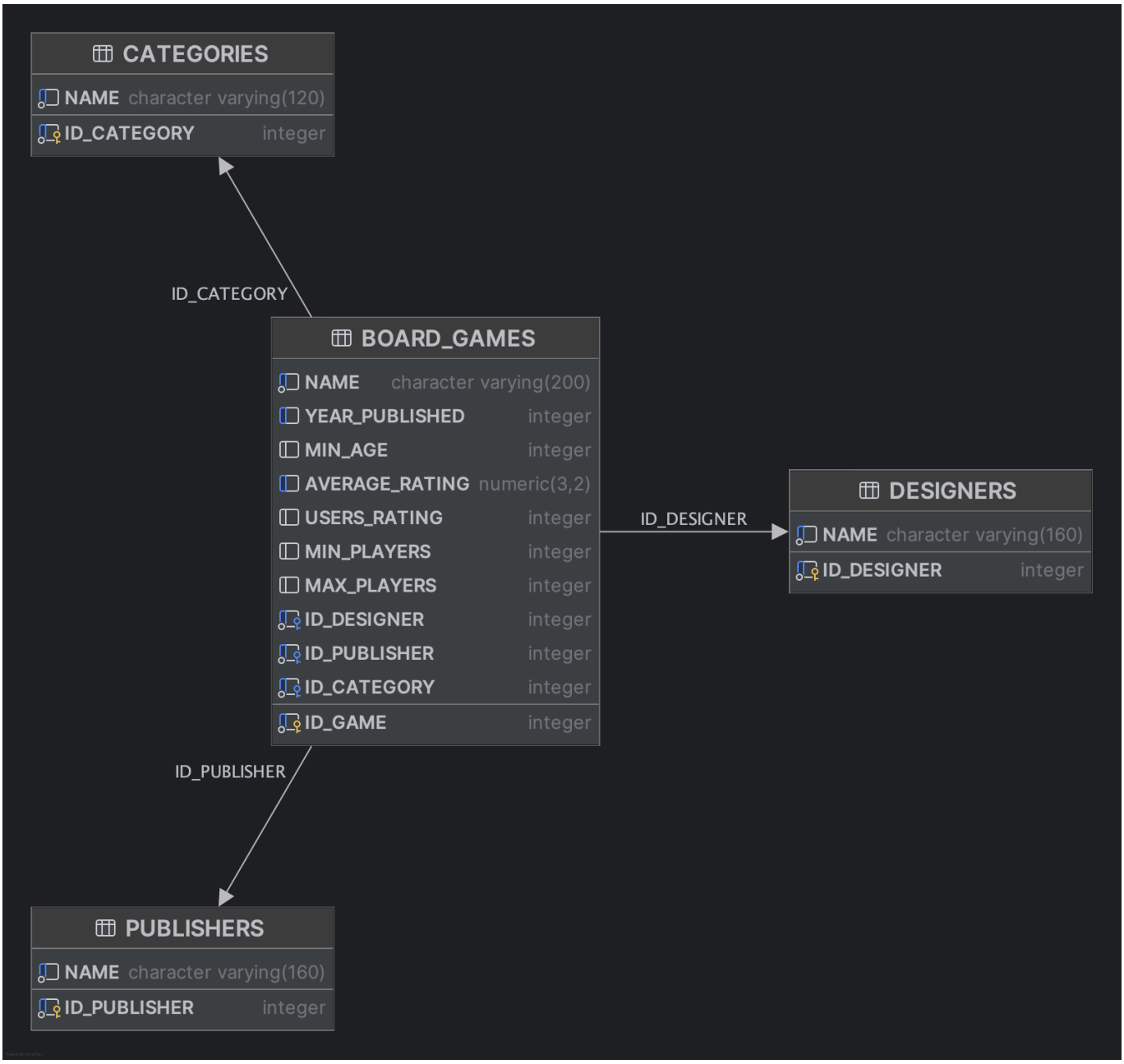
3.1 Inicialización de BD

Implementar `DbInitializer` que:

- Abra una conexión H2 **en memoria** (URL ejemplo: `jdbc:h2:mem:boardgames;DB_CLOSE_DELAY=-1`),
- Ejecute **completo** `sql/ddl_board_games.sql` (desde `classpath:`),
- Cierre recursos correctamente (try-with-resources).

3.1.1 Estructura de la base de datos

Se suministra aquí el DER de la base de datos que se crea a partir del script suministrado.



3.2 Capa de infraestructura

- `DataSourceProvider`: expone un `javax.sql.DataSource` (H2 en memoria) para **JDBC** y **JPA**.
- `LocalEntityManagerProvider`: configura `EntityManagerFactory` con Hibernate (dialecto H2), `hibernate.hbm2ddl.auto=none`, `show_sql` opcional, y **usa el mismo datasource**.

> **Nota:** Es importante mantener los nombres de las clases y paquetes coherentes con los materiales de referencia, ya que se reutilizarán en la etapa del parcial para inicializar y validar la estructura.

3.3 Entidades JPA (mapeos)

Crear entidades para **todas** las tablas del DDL (nombres orientativos):

- `Designer` → `DESIGNERS` (`ID_DESIGNER`, `NAME`)
- `Publisher` → `PUBLISHERS` (`ID_PUBLISHER`, `NAME`)
- `Category` → `CATEGORIES` (`ID_CATEGORY`, `NAME`)
- `BoardGame` → `BOARD_GAMES` (`ID_GAME`, `NAME`, `YEAR_PUBLISHED`, `MIN_AGE`, `AVERAGE_RATING`, `USERS_RATING`, `MIN_PLAYERS`, `MAX_PLAYERS`, FKs…)

Indicaciones:

- Definir `@Id` y estrategia de generación acorde al DDL (secuencias).
- Relaciones en `BoardGame`:
  - `@ManyToOne(optional = false)` a `Designer`, `Publisher`, `Category`.
- Usar `@Column(name = "…")` si el nombre de la columna no coincide con el del campo Java.

3.4 Repositorios (opcional)

- Podés implementar **repos JPA** simples (`findById`, `findAll`, etc.) o usar `EntityManager`/`TypedQuery` directo desde la capa de aplicación. La existencia de `repo/` es **opcional**.

3.5 Main `App.java` (Sugerido)

Se sugiere la construcción de una clase con un método main que al menos compruebe la funcionalidad con las siguientes pruebas:

- Ejecutar `DbInitializer`.
- Construir `EntityManager` vía `LocalEntityManagerProvider`.
- Realizar un **smoke test**: por ejemplo, contar registros en `PUBLISHERS` o `CATEGORIES` y/o crear/leer un registro simple si el seed lo permite.
- Mostrar un **OK/FAIL** claro por consola (p.ej. `"[OK] DB init + JPA mappings verificados"`).

---

4) Guía específica – Validaciones de rango

El DDL define en `BOARD_GAMES` los campos `MIN_PLAYERS`, `MAX_PLAYERS` y `MIN_AGE`. Se sugiere la implementación de un método en la entidad `BoardGame` que permita comprobar si un número determinado de jugadores o una edad cumplen las condiciones del juego.

> **Sugerencia:** implementar en `BoardGame` los métodos:
>
> ```
> public boolean supportsPlayerCount(int players) { /* … */ }
>
> public boolean isSuitableForAges(int[] ages) { /* … */ }
> ```
>
> que retornen `true` si el valor recibido cumple las condiciones establecidas en los campos correspondientes.
>
> *Tip:* Si `MAX_PLAYERS` es `null`, interpretar como sin tope superior (`players >= MIN_PLAYERS`). Si `MIN_PLAYERS` es `null`, no restringir por mínimo. De igual modo para edades.

---

5) Estrategia sugerida

Se sugiere lo siguiente de acuerdo con lo que se se solicitará el día del parcial

- Proyecto Maven compilable con `mvn compile` y ejecutable con `mvn exec:java` cómo está documentado en los materiales y lo hemos trabajado en clase.
- El proyecto debería respetar lo expresado en la presente consigna
    - Ejecutar a partir del **Main** (si los implementa),
    - URL JDBC usada,
    - Dónde está el `persistence.xml` y el `sql/ddl_board_games.sql`,
    - Nota indicando que se utilizarán los proveedores de `DataSource` y `EntityManager`.
- Si se opta por otra alternativa, algunos elementos podrían alterar el funcionamiento de los componentes documentados.

Sobre el uso de git u otros repositorios (Disclaimer)

No se requiere el uso de un repositorio específico para la contención temporal de lo vertido en la construcción del proyecto a partir del presente pre-enunciado, sin embargo:

- Los docentes de la cátedra no se hacen responsables, ni tienen la obligación de brindar solución ante la pérdida del trabajo realizado.
- Los docentes de la cátedra no tienen responsabilidad de que el entorno de trabajo funcione ni que cargue el pre-enunciado que haya generado.
- La cátedra no se hace responsable el día de la evaluación sobre elementos del entorno de trabajo ni de las dependencias asociadas a este... esto es parte de la evaluación.

> **Recordatorio**: todo lo opcional busca que, el día del parcial, puedas enfocarte exclusivamente en **parseo de CSV** y **consultas**.

Éxitos a todos en el desarrollo del parcial