Ejercicio 1ª Evaluación – Acceso a Datos (Ficheros JSON y BBDD H2 con Spring Data JPA)

1. Descripción general del ejercicio

Este ejercicio de evaluación integra **tres bloques de trabajo**:

1. Este ejercicio pretende que demuestres si has asumido los conceptos de trabajo conn ficheros (Json) y bbdd en Java (Spring + jpa + h2) con la ayuda de las tareas que has desarrollado en clase. Para eso debes de:

- **Persistencia en ficheros JSON (Roles)** – *(hasta 3 puntos)*
- **Persistencia en BBDD H2 con Spring Data JPA (Alumnos y Roles)** – *(hasta 6 puntos)*
- **Cuestionario teórico tipo test** – *(hasta 1 punto)*

Puntuación total del ejercicio: **10 puntos**.

---

2. Bloque A – Ficheros JSON (Roles) – Máximo 3 puntos

En esta parte se trabaja con la clase de dominio `Rol` y un repositorio basado en **ficheros JSON** (por ejemplo, `RolJsonFileRepository`). Debes de:

- Serializar/deserializar objetos `Rol` y almacenar y modificar ficheros.

2.1. Operaciones a implementar

Cada estudiante debe ser capaz de implementar un **CRUD completo** sobre roles en fichero JSON. En concreto:

1. **Leer todos los roles desde un fichero JSON**
   - Método típico: `List<Rol> findAll()`
2. **Buscar un rol por id**
   - Método típico: `Optional<Rol> findById(Long id)`
3. **Buscar un rol por nombre**
   - Método típico: `Optional<Rol> findByNombre(String nombre)`
4. **Crear/actualizar roles y volcarlos al fichero**
   - Método típico: `Rol save(Rol rol)`
5. **Borrar roles del fichero**
   - Método típico: `void deleteById(Long id)`
6. **Contar cuántos roles hay almacenados**
   - Método típico: `long count()`

> **Optional** se puede sustituir por List, aunque debes modificar la interfaz y los test.

Cada uno de estos 6 métodos vale **0,5 puntos**, con la siguiente escala:

- **0 puntos** → No funciona.
- **0,3 puntos** → Funciona correctamente.
- **0,4 puntos** → Funciona + está documentado.
- **0,5 puntos** → Funciona + está documentado + es óptimo (código limpio y eficiente).

2.2. Rúbrica detallada – Ficheros JSON (Roles)

Método / operación	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Funciona + documentado	0,5 pts – Funciona + documentado + óptimo
`List<Rol> findAll()`	Leer todos los roles desde un fichero JSON.	No compila o devuelve vacío sin sentido.	Devuelve correctamente la lista.	+ Javadoc y nombres adecuados.	+ Código limpio, buen manejo de errores.
`Optional<Rol> findById(Long id)`	Buscar un rol por id.	No funciona o lanza errores.	Devuelve correctamente el rol o vacío.	+ Documentación clara.	+ Uso idiomático de `Optional`, sin condicionales innecesarios.
`Optional<Rol> findByNombre(String nombre)`	Buscar un rol por nombre.	Ignora el nombre o lanza errores.	Devuelve el rol correcto o vacío.	+ Comentarios explicativos.	+ Uso de streams, sin duplicar lógica.
`Rol save(Rol rol)`	Crear/actualizar roles y guardar.	No guarda o corrompe JSON.	Guarda/actualiza correctamente.	+ Documentación sobre id y flujo.	+ Manejo de ids, JSON consistente y eficiente.
`void deleteById(Long id)`	Borrar roles del fichero.	No borra o lanza errores.	Elimina solo el rol indicado.	+ Comentarios sobre casos inexistentes.	+ Implementación clara y eficiente.
`long count()`	Contar roles almacenados.	No funciona o lanza errores.	Devuelve número real de roles.	+ Documentación sobre conteo.	+ Implementación directa y sin redundancias.

Nota: Máximo total del bloque = **3 puntos.**

---

3. Bloque B – BBDD H2 + Spring Data JPA (Alumnos y Roles) – Máximo 6 puntos

En esta parte se trabaja con una base de datos en memoria **H2**, gestionada mediante **Spring Boot + Spring Data JPA**.

Modelos principales:

- `Alumno` / `AlumnoEntity` (tabla `alumnos`)
- `Rol` / `RolEntity` (tabla `roles`)

Relación entre ellos:

- Un **Alumno** tiene un **Rol** (`ManyToOne`)
- Un **Rol** tiene muchos **Alumnos** (`OneToMany`)

3.1. Mapeo de entidades JPA (1 operación)

Operación	Clases / Contexto	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Funciona + documentado	0,5 pts – Funciona + documentado + óptimo

Operación	Clases / Contexto	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Funciona + documentado	0,5 pts – Funciona + documentado + óptimo
Mapeo JPA de `AlumnoEntity` y `RolEntity`	Entidades JPA (`AlumnoEntity`, `RolEntity`)	Configurar correctamente las tablas y relaciones.	La app no arranca o relaciones incorrectas.	Tablas correctas, relación funcional.	+ Comentarios y Javadoc.	+ Buenas prácticas JPA (`@ManyToOne`, `@OneToMany`, equals/hashCode, restricciones).

3.2. Adaptador JPA `AlumnoRepositoryJpaAdapter` (6 operaciones)

Operación	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Documentado	0,5 pts – Óptimo
`List<Alumno> findAll()`	Lista de alumnos.	Falla o devuelve vacío.	Devuelve lista correcta.	+ Documentado.	+ Código limpio, sin duplicados.
`Optional<Alumno> findById(Long id)`	Buscar alumno.	Falla o lanza errores.	Devuelve correcto o vacío.	+ Documentado.	+ Uso idiomático de `Optional`.
`Alumno save(Alumno alumno)`	Guardar/actualizar.	No persiste o rompe relación.	Guarda bien.	+ Documentado.	+ Mapeos claros (`toDomain`, `toEntity`).
`boolean existsByEmail(String email)`	Comprobar existencia.	Falla o siempre igual.	Devuelve correcto.	+ Documentado.	+ Usa métodos derivados eficientemente.
`void deleteById(Long id)`	Eliminar por id.	No borra.	Elimina correctamente.	+ Documentado.	+ Implementación directa y limpia.
`long count()`	Contar registros.	Falla o incorrecto.	Devuelve total real.	+ Documentado.	+ Uso directo de `jpa.count()`.

3.3. Repositorio JPA `RolJpaRepository` (evaluación independiente)

Operación	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Documentado	0,5 pts – Óptimo
`RolEntity save(RolEntity rol)`	Guarda un nuevo rol y genera id.	No guarda o lanza error.	Guarda correctamente y genera id.	+ Documentado.	+ Mapeo y validaciones coherentes, uso correcto de transacciones.
`Optional<RolEntity> findById(Long id)`	Buscar rol por id.	Falla o devuelve vacío siempre.	Devuelve correcto o vacío.	+ Documentado.	+ Implementación clara y uso idiomático de `Optional`.

Operación	Descripción	0 pts – No funciona	0,3 pts – Funciona correctamente	0,4 pts – Documentado	0,5 pts – Óptimo
`List<RolEntity> findAll()`	Lista de roles.	Falla o devuelve vacío.	Devuelve lista correcta.	+ Documentado.	+ Código limpio, ordenado, sin redundancias.
`void deleteById(Long id)`	Eliminar rol.	No borra o lanza error.	Borra correctamente.	+ Documentado.	+ Implementación clara y eficiente.
`long count()`	Contar roles.	Falla o devuelve valor erróneo.	Devuelve total correcto.	+ Documentado.	+ Uso directo de `count()` optimizado.
`Optional<RolEntity> findByNombre(String nombre)` *(opcional)*	Buscar por nombre.	No implementado o incorrecto.	Devuelve correcto o vacío.	+ Documentado.	+ Implementación eficiente (query derivada o JPQL).

4. Bloque C – Test teórico tipo test – Máximo 1 punto

Evalúa conocimientos de:

- Acceso a datos y persistencia.
- JSON, XML y serialización con Jackson.
- Spring Data JPA y relaciones entre entidades.
- Manejo de errores y validaciones.