Examen: CRUD con JPA (H2) + MongoDB

En este ejercicio se pide que:

- Modelo de datos relacional con **JPA** y BBDD **H2**.
- Modelo documental con **MongoDB**.
- Relación lógica entre ambas BBDD.
- Implementación de un **CRUD** básico en capas (controller (opcional), service, repository).
- **Capa de servicio** para cada repositorio.

> Puedes hacer uso del proyecto `spring-rest-producto-resuelto-clase` o completar el que se adjunta con el nombre `tiendaonline-exam`.

1. Enunciado (resumen)

Implementar una aplicación Spring Boot que gestione los pedidos de una tienda online con:

1. **BBDD relacional H2 (JPA)**:
    - Tabla `CLIENTE` (ID, NOMBRE, EMAIL).
    - int id;
    - String nombre;
    - String email;
    - List pedidos;
    - Tabla `PEDIDO` (ID, FECHA, ESTADO, CLIENTE_ID).
    - int id;
    - String estado;
    - cliente;
    - Relaciones:
        - `CLIENTE 1 --- N PEDIDO`.
2. **BBDD No relacional MongoDB**:
    - Colección `cliente_detalles` que complemente a `CLIENTE`, enlazada por el campo `clienteId`.
    - Las propiedades son las siguiente:
    - String id;
    - int clienteId;
    - String telefono;
    - String notasInternas;
3. **Operaciones CRUD mínimas**:
    - CRUD completo de `Cliente`.
    - Alta de `Pedido` para un `Cliente` existente.
    - Consulta de un `Cliente` que devuelva también sus detalles desde MongoDB, si existen.
4. **Requisitos extra en este proyecto**:

- Una **capa de servicio** específica por cada repositorio:
  - `ClienteService`
  - `PedidoService`
  - `ClienteDetallesService`

2. Estructura del proyecto

```
tiendaonline-exam
├── pom.xml
├── README.md
└── src
    ├── main
    │   ├── java
    │   │   └── com
    │   │       └── tiendaonline
    │   │           ├── TiendaOnlineApplication.java
    │   │           ├── controller
    │   │           │   ├── ClienteController.java (Para que verifiques el
correcto funcionamiento a mano)
    │   │           │   └── PedidoController.java
    │   │           ├── service
    │   │           │   ├── ClienteService.java
    │   │           │   ├── PedidoService.java
    │   │           │   └── ClienteDetallesService.java
    │   │           ├── repository
    │   │           │   ├── ClienteRepository.java
    │   │           │   ├── PedidoRepository.java
    │   │           │   └── ClienteDetallesRepository.java
    │   │           └── model
    │   │               ├── Cliente.java
    │   │               ├── Pedido.java
    │   │               └── ClienteDetalles.java
    │   └── resources
    │      └── application.properties
    │
    └── test
        └── java
            └── (No hay test porque si no tengo que crear las estructuras)
```

3. Cómo ejecutar

1. Asegúrate de tener configurar correctente la bbdd
   - Un MongoDB en marcha (por defecto `mongodb://localhost:27017/tiendaonline`).
2. Ejecuta la aplicación:

```
mvn spring-boot:run
```

3. La API quedará disponible (por defecto) en: `http://localhost:8080`. Si implementas la capa controller

Endpoints principales:

- `GET /clientes`
- `GET /clientes/{id}`
- `POST /clientes`
- `PUT /clientes/{id}`
- `DELETE /clientes/{id}`
- `POST /clientes/{id}/detalles`
- `POST /pedidos/cliente/{clienteId}`
- `GET /pedidos/{id}`

4. Qué se evalúa con este proyecto

- Correcta **modelización relacional** (tablas, tipos, claves, relaciones).
- Uso adecuado de **JPA** y anotaciones de entidades.
- Correcta relación lógica con **MongoDB** mediante `clienteId`.
- Separación en capas (**controller**, **service**, **repository**, **model**).
- Uso de **capa de servicio** como intermediaria entre controladores y repositorios.
- Buenas prácticas de tests unitarios: verificación de:
  - Orden de resultados.
  - Listas vacías.
  - Parámetros nulos que deben lanzar excepciones.
- Correcto funcionamiento de cada uno de los métodos que se definen en la capa de servicios.

> Si no funciona el método no se evalua la parte anterior.