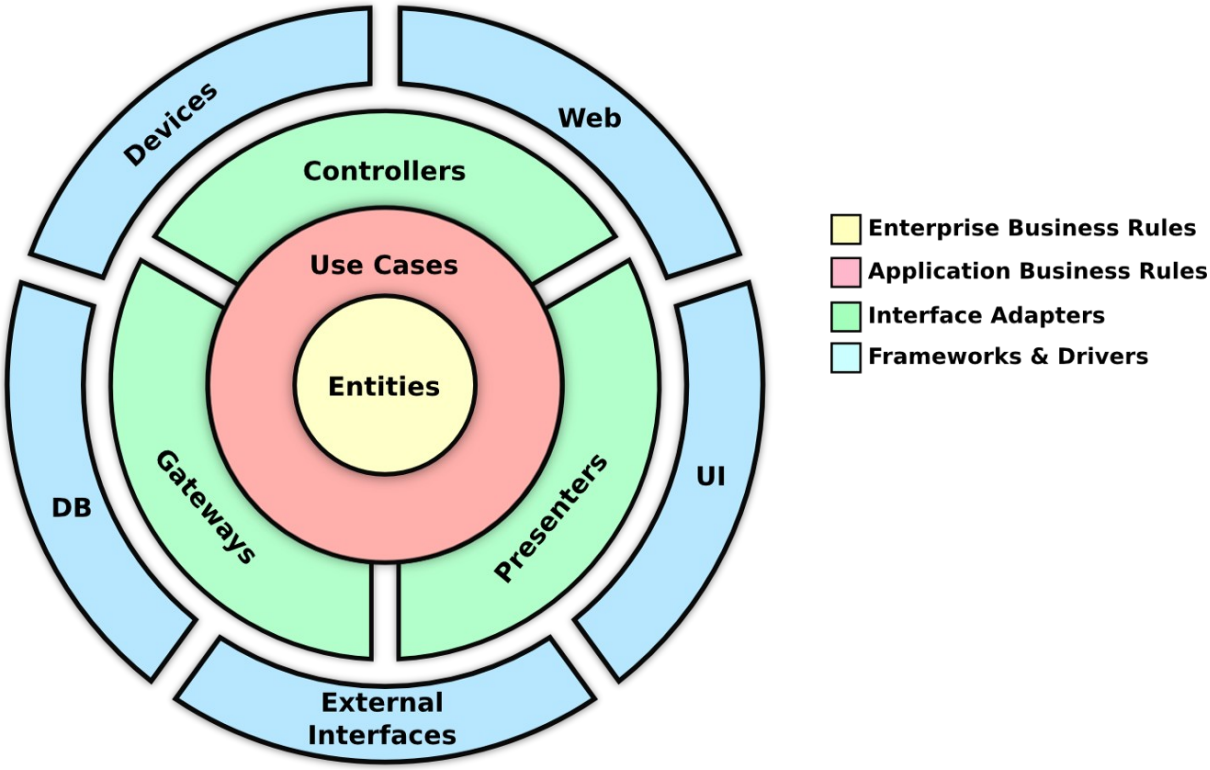
1.1. Problema:

El departamento de informática tiene libros sin digitalizar. Se quiere realizar un CRUD de libro que permita mediante fotografía, OCR e IA digitalizar la información para realizar un volcado en la BBDD de Séneca.

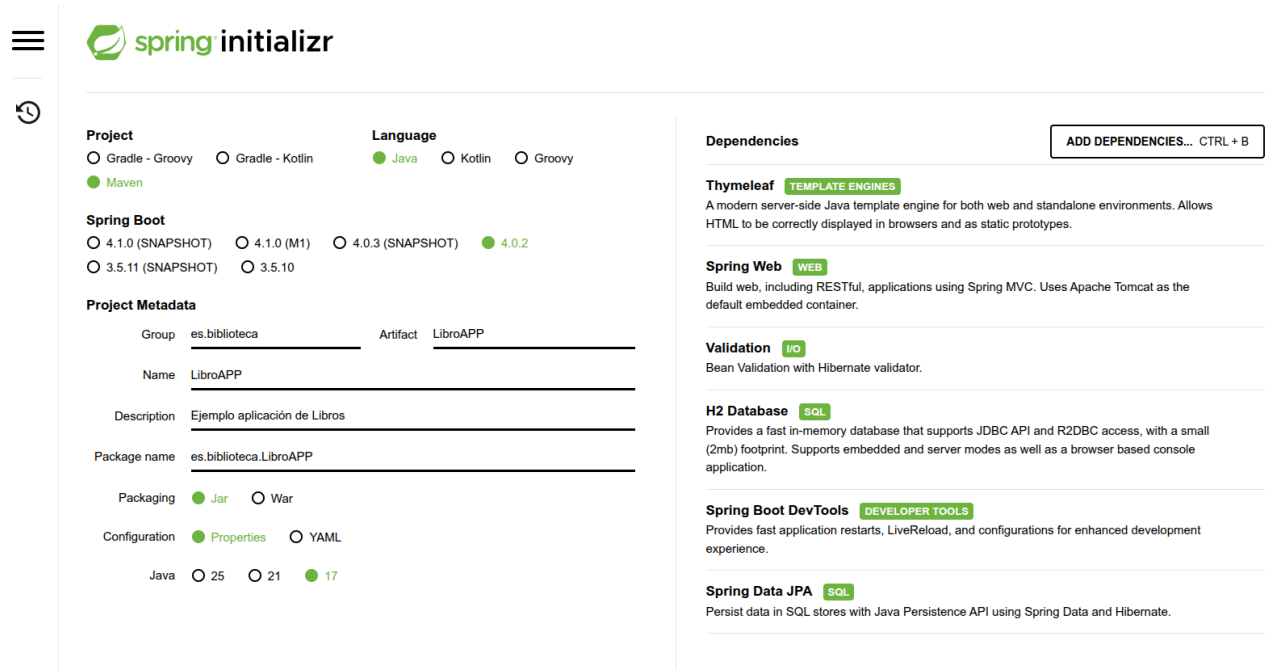


1.2.Posible solución:

Crear aplicación MVC sencilla de CRUD de Libro que permita tras subir una imagen de la portada y la contraportada. Realizar OCR en alguna IA open source que permita persistir la información del libro.

2. Crear proyecto. Pasos a seguir:

- Crear proyecto con Spring Boot (https://start.spring.io/) y agregar dependencias con Spring Boot.
  - Thymeleaf para crear vistas con el Controlador
  - Validation para las validaciones de las Entidades
  - Spring Web que permita usar Aparche Tomcat como contenedor embebido por defecto para crear aplicaciones Spring MVC
  - H2 Database para desarrollo inicial.
  - Spring Data JPA para realizar el mapeo relacional de Entidades con la Base de datos.
  - Spring Boot Dev Tools para fácilitar el desarrollo.
- Gestión de versiones con Git y Github.
- (Diagrama estático) Modelo de Dominio de la App

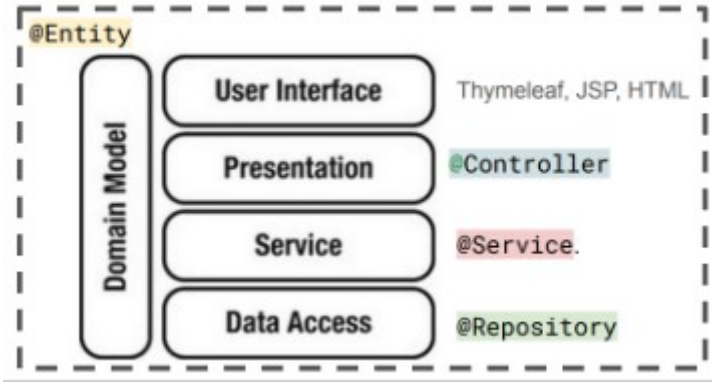


2.1. Fichero de configuración

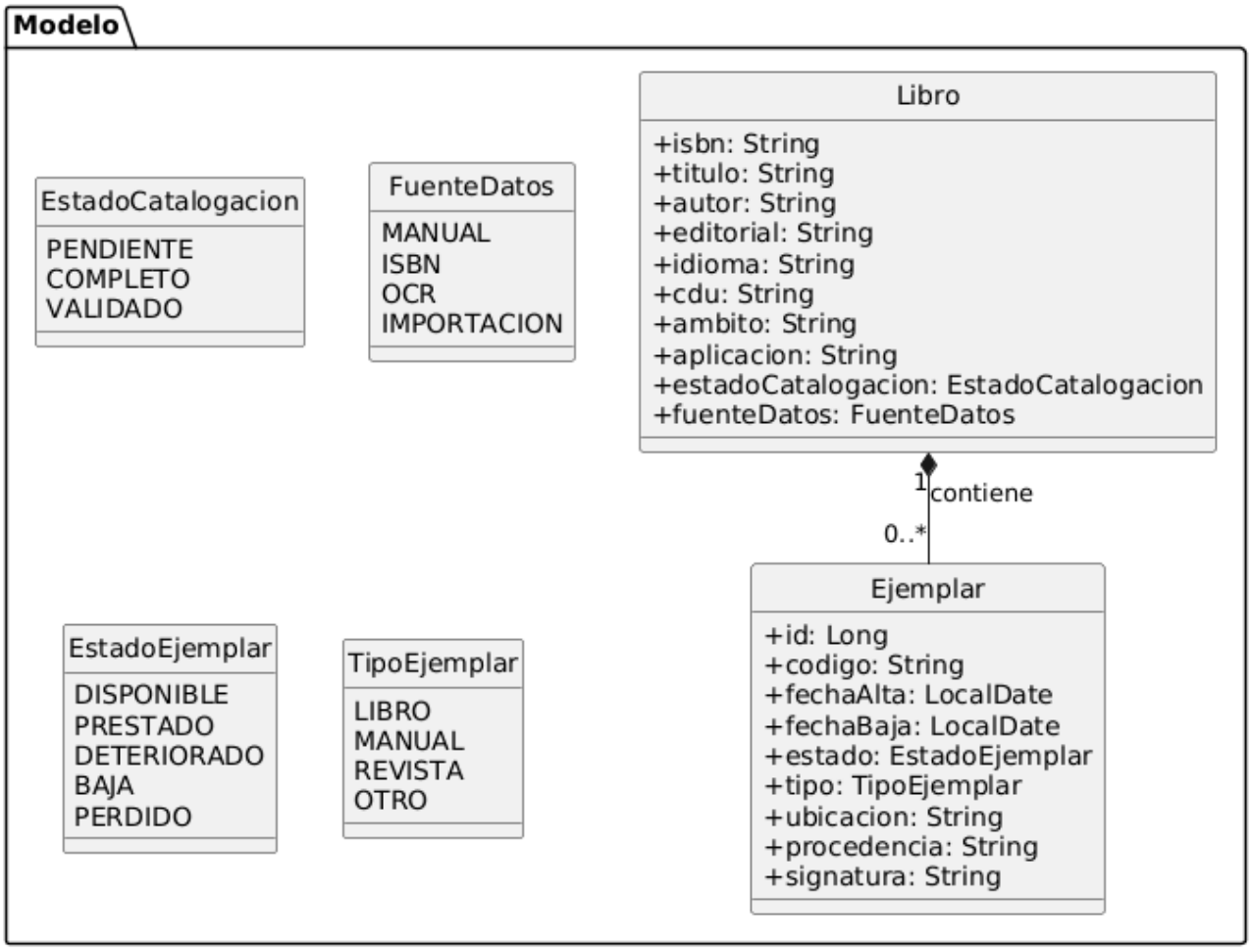
Actualizar el archivo **application.properties**

```
spring.application.name=LibroAPP
server.port=9092

spring.h2.console.enabled=true
spring.h2.console.path=/h2

spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.hibernate.ddl-auto=create-drop
```

3. Modelo de dominio de la aplicación



Modelo de Dominio - Libro y Ejemplar (MVC + Spring Boot)



Un **Libro** puede tener ningún o muchos **Ejemplares**
No puede existir Ejemplar sin libro.

3.1. Entidad seleccionada para CRUD inicial.

```
package es.biblioteca.LibroAPP.modelo.entidad;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.validation.constraints.NotBlank;

@Entity
public class Libro {
	@Id
	@GeneratedValue(strategy =GenerationType.AUTO)
	private Integer id;

	@NotBlank(message = "Nombre no puede estar en blanco")
	private String titulo;

	@NotBlank(message = "isbn no puede estar en blanco")
	@Column(nullable = false, unique = true)
	private String isbn;

	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public String getIsbn() {
		return isbn;
	}
	public void setIsbn(String isbn) {
		this.isbn = isbn;
	}


}
```

3.2. Repositorio para Entidad seleccionada

```
package es.biblioteca.LibroAPP.repositorio;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import es.biblioteca.LibroAPP.modelo.entidad.Libro;

@Repository
public interface LibroRepository extends JpaRepository<Libro, Integer>{

    boolean existsByIsbn(String isbn);
    Optional<Libro> findByIsbn(String isbn);
    List<Libro> findByTituloContainingIgnoreCase(String titulo);
}
```

3.3. Servicio y su implementación

En Spring Boot es habitual definir el servicio en **dos piezas**:

- **Interfaz (`LibroService`)**: define *qué operaciones ofrece* el servicio.
- **Implementación (`LibroServiceImpl`)**: define *cómo se realizan* esas operaciones.

3.3.1. Ventajas prácticas

✅ A) Cambiar la base de datos sin tocar el Controller

El Controller depende de `LibroService` (interfaz), no de JPA.
Si mañana cambias de H2 → MySQL → PostgreSQL, o cambias la estrategia de persistencia, el Controller no cambia.

✅ B) Facilita testear

Al depender de una interfaz, en tests puedes sustituir el servicio por un "mock" y probar el Controller sin base de datos.

✅ C) Sitio correcto para reglas de negocio

Validaciones "de negocio" van en el servicio:

- normalizar ISBN (quitar guiones/espacios)
- evitar duplicados
- decidir errores y mensajes
- transacciones (`@Transactional`)

El Controller debe centrarse en web + vistas.

✅ D) Escalar el proyecto sin reestructurar

Cuando el sistema crece (Ejemplares, exportación a Séneca, OCR, autocompletado…), tener esta separación evita que el proyecto se convierta en un “spaghetti”.

3.3.2. Interfaz

```
public interface LibroService {
        // CRUD
    Libro crear(Libro libro);
    Libro actualizar(Integer id, Libro libro);
    Optional<Libro> buscarPorId(Integer id);
    List<Libro> listar();
    void eliminar(Integer id);

    // Operaciones por ISBN (muy útiles para tu caso)
    Optional<Libro> buscarPorIsbn(String isbn);
    boolean existeIsbn(String isbn);

    // Búsqueda simple para el listado (q)
    List<Libro> buscar(String q);
}
```

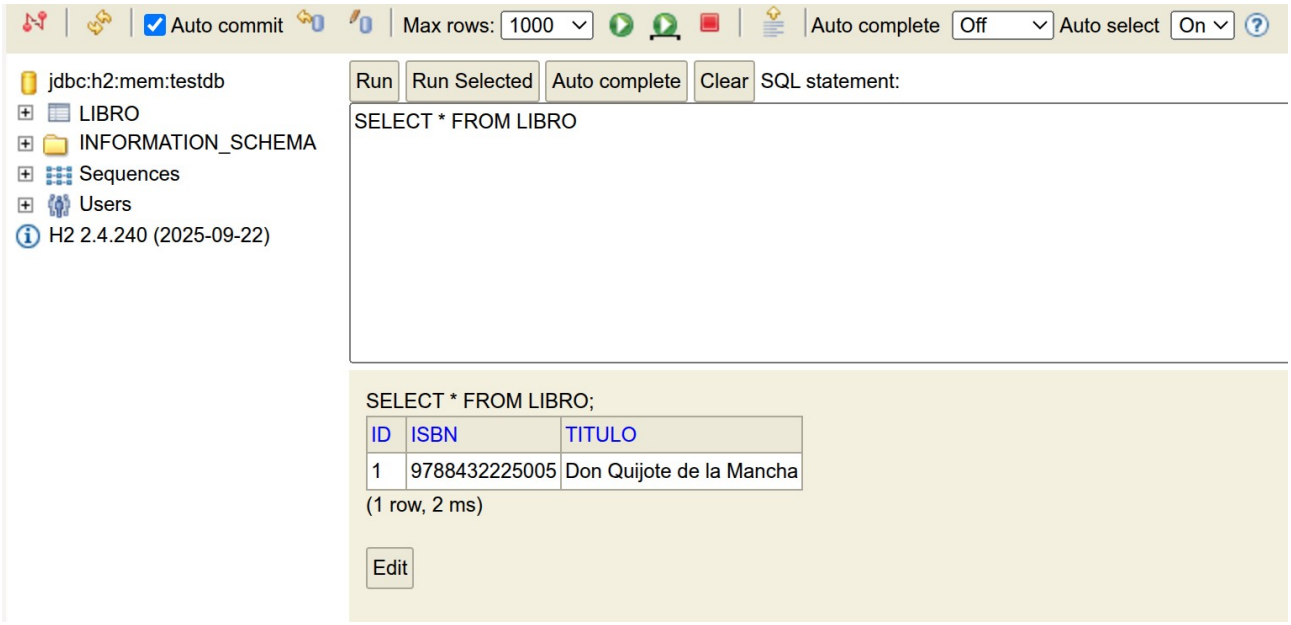
3.3.3. Implementación de la Intefaz del Servicio.

```
@Service
     public class LibroServiceImpl implements LibroService {

         private final LibroRepository repositorio;
         public LibroServiceImpl(LibroRepository repositorio) {
             this.repositorio = repositorio;
         }

         @Override
         public Libro crear(Libro libro) {
             normalizar(libro);
             // Evitar duplicado por ISBN
             if (repositorio.existsByIsbn(libro.getIsbn())) {
                 throw new IllegalArgumentException("ISBN ya
existe");
             }

             return repositorio.save(libro);
         }
```

3.3.4. Agregar datos para probar servicio.

```
@Component
public class DatosFakeDesarrollo  implements CommandLineRunner{

     @Autowired
     private LibroService servicio;


     @Override
     public void run(String... args) throws Exception {
          Libro libro = new Libro();
          libro.setTitulo("Don Quijote de la Mancha");
          libro.setIsbn("9788432225005");
          servicio.crear(libro);

     }

}
```

Comprobación http://localhost:9092/h2



Anexo

4.1. Entidad : `Libro` (obra / registro bibliográfico)

`isbn`

- **Descripción:** Identificador bibliográfico del libro.
- **Tipo:** `String` (VO `ISBN`)
- **Formato estándar: ISBN-13** (13 dígitos), puede venir con guiones pero se almacena **normalizado sin guiones**.
    - Ej: `9780134685991`
- **Restricciones:**
    - **Único**.
    - Validación de dígitos y **check digit** de ISBN-13.
    - **Permitido "interno" temporal** si no existe ISBN: `SINISBN-000001` (patrón propio).
- **Regla recomendada:** guardar además `isbnOriginal` si quieres conservar el texto tal como venía (opcional).

`titulo`

- **Descripción:** Título principal.
- **Tipo:** `String`
- **Restricciones:** obligatorio, `@NotBlank`, longitud recomendada `1..255`.
- **Normalización:** trim, espacios múltiples → uno.

`autor`

- **Descripción:** Autor/es tal como se muestran.
- **Tipo:** `String`
- **Formato recomendado:** `"Apellido, Nombre; Apellido, Nombre"` (separador `;` para múltiples).
- **Restricciones:** opcional al inicio; longitud recomendada `0..255`.

`editorial`

- **Descripción:** Editorial / sello.
- **Tipo:** `String`
- **Restricciones:** opcional; `0..120` recomendado.

`idioma`

- **Descripción:** Idioma del libro.
- **Tipo:** `String`
- **Formato estándar: ISO 639-1** (2 letras) o ISO 639-2 (3 letras).
    - Recomendado: `es`, `en`, `fr`…
- **Restricciones:** opcional; si se rellena, validar contra patrón `^[a-z]{2,3}$`.

`Cdu (Campo Séneca)`

- **Descripción:** Clasificación Decimal Universal.
- **Tipo:** `String`
- **Formato recomendado:** texto CDU tal cual (puede incluir puntos, dos puntos, paréntesis).

    - Ej: `004.4`, `004.7:37`
- **Restricciones:** opcional; longitud recomendada `0..30`.

`ambito`

- **Descripción:** Ámbito o categoría interna (p.ej. “DAW”, “SMR”, “Bachillerato”).
- **Tipo:** `String` (o enum si lo tenéis cerrado)
- **Restricciones:** opcional; `0..60`.

`aplicacion`

- **Descripción:** Uso/materia interna (p.ej. “Programación”, “Redes”, “Seguridad”).
- **Tipo:** `String` (o enum)
- **Restricciones:** opcional; `0..60`.

`estadoCatalogacion`

- **Descripción:** Estado del registro bibliográfico.
- **Tipo:** `Enum`
- **Valores:** `PENDIENTE | COMPLETO | VALIDADO`
- **Restricciones:** obligatorio, por defecto `PENDIENTE`.
- **Regla:** `VALIDADO` solo si los campos mínimos para Séneca están completos.

`fuenteDatos`

- **Descripción:** Origen principal del registro.
- **Tipo:** `Enum`
- **Valores:** `MANUAL | ISBN | OCR | IMPORTACION`
- **Restricciones:** obligatorio, por defecto `MANUAL`.

4.2. Entidad: `Ejemplar` (copia física / inventario)

`id`

- **Descripción:** Identificador interno técnico.
- **Tipo:** `Long` (VO `EjemplarId`)
- **Restricciones:** autogenerado.

`codigo`

- **Descripción:** Código interno único del ejemplar (pegatina/QR).
- **Tipo:** `String` (VO `CodigoEjemplar`)
- **Formato recomendado (propio):**
    - `IESCAMAS-INF-000001` (prefijo centro + depto + secuencia)
    - o `INF-DAW-2026-0001`
- **Restricciones:**
    - **Único** y **obligatorio**.
    - Patrón recomendado: `^[A-Z0-9-]{6,30}$` (ajústalo a vuestro estilo).
- **Regla:** no reutilizar códigos dados de baja.

`fechaAlta`

- **Descripción:** Fecha en que el ejemplar entra al inventario.
- **Tipo:** `LocalDate` (VO `Fecha`)

- **Formato estándar: ISO 8601** YYYY-MM-DD
- **Restricciones:** recomendado; si vacío, se puede rellenar con “hoy” al crear.

`fechaBaja`

- **Descripción:** Fecha de baja del ejemplar (si aplica).
- **Tipo:** `LocalDate`
- **Formato:** ISO 8601
- **Restricciones / reglas:**
  - `fechaBaja >= fechaAlta`
  - si `estado = BAJA`, `fechaBaja` debería existir.

`tipo`

- **Descripción:** Tipo de material.
- **Tipo:** `Enum`
- **Valores recomendados:** `LIBRO | MANUAL | REVISTA | OTRO`
- **Restricciones:** obligatorio, por defecto `LIBRO`.

`estado`

- **Descripción:** Estado del ejemplar.
- **Tipo:** `Enum`
- **Valores:** `DISPONIBLE | PRESTADO | DETERIORADO | BAJA | PERDIDO`
- **Restricciones:** obligatorio, por defecto `DISPONIBLE`.
- **Reglas recomendadas:**
  - BAJA implica `fechaBaja` informada.
  - PRESTADO normalmente lo determinará el módulo de préstamos (si existe).

`ubicacion`

- **Descripción:** Ubicación física (armario, aula, estantería).
- **Tipo:** `String`
- **Formato recomendado:** código corto + descripción
  - Ej: `AULA-2.12 / ARM-01 / BALDA-3`
- **Restricciones:** opcional; `0..80`.

`procedencia`

- **Descripción:** Origen del ejemplar (compra, donación, programa…).
- **Tipo:** `String`
- **Restricciones:** opcional; `0..80`.

`signatura`

- **Descripción:** Signatura/topográfico interno (si lo usáis).
- **Tipo:** `String`
- **Formato recomendado:** basado en CDU + iniciales:
  - Ej: `004.4 MAR pro`
- **Restricciones:** opcional; `0..40`.

Relación `Ejemplar -> Libro`

- **Descripción:** cada ejemplar pertenece a **un** libro.
- **Cardinalidad:** `Libro 1 ── 0..* Ejemplar`
- **Restricción:** `Ejemplar.libro` obligatorio (FK a `isbn`).

Reglas de integridad

- **`isbn`** único en `Libro`.
- **`codigo` único** en `Ejemplar`.
- No permitir **`Ejemplar`** sin **`Libro`**.
- `fechaBaja` solo si `estado` en `{BAJA, PERDIDO}` (según vuestra política).
- `estadoCatalogacion=VALIDADO` exige como mínimo:
    - `isbn` (o interno), `titulo` y al menos **uno** entre `autor/editorial`.