



# Documentacion\_Libros\_Y\_Autores

## Documentación del Proyecto - Libros y Autores

### 1. Introducción

Este proyecto es una aplicación desarrollada en **Java** usando el metaframework **Spring Boot**. Su propósito es gestionar información relacionada con  **libros** y  **autores**, permitiendo operaciones básicas de tipo CRUD (crear, leer, actualizar y eliminar).

---

### 2. Descripción del proyecto



La aplicación está organizada en varias capas:

- Controladores ( `controller` )
- Servicios ( `service` )
- Repositorios ( `repository` )
- Modelos ( `model` )

Esta estructura sigue buenas prácticas de arquitectura y separación de responsabilidades para mantener el código limpio y mantenible.

---

### 3. Tecnologías utilizadas

-  Java 17
  -  Spring Boot
  -  Maven
  -  JPA / Hibernate
  -  Git
- 

### 4. Estructura del proyecto

```
libros_y_autores/  
├─ controller/ # Controladores REST  
├─ service/ # Lógica de negocio  
├─ repository/ # Acceso a datos  
├─ model/ # Entidades Autor y Libro  
├─ resources/ # Configuración (application.properties)  
└─ test/ # Pruebas del sistema
```

---

## ✓ 5. Funcionalidades principales

- 📋 Listar libros
- + Añadir libros nuevos
- 🔗 Asociar autores a libros
- ✎ Editar libros
- 🗑 Eliminar libros

---

## 🧠 División de trabajo

### 📁 Izan — Encargado de la entidad **Libro**

Responsabilidades:

- Crear la clase `Libro.java` en el paquete `model`, incluyendo el campo `autor` como relación `@ManyToOne`.
- Desarrollar `LibroRepository.java`.
- Implementar `LibroService.java` con la lógica de negocio: creación, listado, filtrado, actualización, eliminación.
- Crear `LibroController.java` con los siguientes endpoints:
  - `GET /api/v1/libros`
  - `GET /api/v1/libros/{id}`
  - `POST /api/v1/libros` (vinculando con un `Autor`)
  - `PUT /api/v1/libros/{id}`
  - `DELETE /api/v1/libros/{id}`
  - `GET /api/v1/libros/buscar` (soporte para filtros: `titulo`, `anio`, y ordenamiento `sortBy` y `order`)

- Probar los endpoints de libros en Postman.
- Documentar en el README la parte de uso de libros.

## **Diego — Encargado de la entidad Autor**

Responsabilidades:

- Crear la clase `Autor.java` en el paquete `model`, con una relación `@OneToMany(mappedBy = "autor")` hacia libros.
  - Desarrollar `AutorRepository.java`.
  - Implementar `AutorService.java` con la lógica para listar, crear y ver autores con sus libros.
  - Crear `AutorController.java` con los siguientes endpoints:
    - `GET /api/v1/autores`
    - `GET /api/v1/autores/{id}`
    - `POST /api/v1/autores`
  - Validar que se muestren correctamente los libros al consultar un autor.
  - Documentar en el README la parte de uso de autores.
- 

## **Notas para el equipo**

- Ambos deben hacer commits con su propio usuario en GitHub.
  - Se recomienda que cada uno trabaje en su **propia rama** (`libros` e `autores`) y luego se haga el merge a `main`.
  - Cada uno debe subir su parte con una breve explicación en el README si es posible.
  - Hacer pruebas cruzadas: tú pruebas los endpoints de Diego y él los tuyos.
- 

## **6. Cómo ejecutar el proyecto**

1. Ejecuta el proyecto.
  2. Abre POSTMAN y prueba los métodos `GET`, `POST`, `PUT` y `DELETE`.
  3. En PHPMyAdmin deberías ver la modificación en la base de datos
- 

## **7. Autor**

## Izan y Diego



Estudiantes de desarrollo web y desarrolladores del proyecto 🚀