Ejercicios con Controladores y Servicios

🌼 Ejercicio 3: Agenda de tareas

🎯 **Objetivo:**
Permitir a un usuario ver una lista de tareas, agregar nuevas y marcarlas como completadas.
Separar la lógica en una capa de servicio. **Resolver utilizando TDD.**

📝 **Requisitos:**

1. Vista tareas.html**:**

- Mostrar una lista de tareas con:
  - Nombre de la tarea
  - Estado: pendiente o completada
- Cada tarea debe tener un botón para marcarla como completada
- Un formulario para agregar nuevas tareas:
  - Campo de texto para el nombre
  - Botón “Agregar tarea”

2. Clase Tarea **(modelo):**

```
public class Tarea {
    private Long id;
    private String nombre;
    private boolean completada;
}
```

3. Servicio TareaService**:**

- Interfaz con métodos:
  - List<Tarea> obtenerTodas()
  - void agregar(Tarea tarea)
  - void marcarComoCompletada(Long id)
- Implementación TareaServiceImpl con lista en memoria
- Agregar **tests unitarios** para todos los métodos del servicio (por ejemplo, usando JUnit y Mockito)

4. Controlador TareaController**:**

- @GetMapping("/tareas") → muestra la lista
- @PostMapping("/tareas") → agrega una tarea
- @PostMapping("/tareas/{id}/completar") → marca como completada

💡 **Tips:**

- Usar @Service en TareaServiceImpl
- Inyectar el servicio en el controlador con @Autowired o constructor
- Usar th:each, th:if, th:action en la vista Thymeleaf
- Escribir primero los tests para el servicio antes de implementar la lógica

🌼 Ejercicio 4: Buscador de libros

🎯 **Objetivo:**
Implementar una búsqueda de libros por título o autor usando una capa de servicio.
Resolver utilizando TDD.

📝 **Requisitos:**

1. Vista buscar.html**:**

- Formulario de búsqueda:
  - Campo de texto
  - Botón de búsqueda
- Mostrar resultados (si hay):
  - Tabla con columnas: título, autor, año

2. Clase Libro **(modelo):**

```
public class Libro {
    private Long id;
    private String titulo;
    private String autor;
    private int anio;
}
```

3. Servicio LibroService**:**

- Interfaz con:
  - List<Libro> buscar(String texto)
- Implementación LibroServiceImpl:
  - Lista hardcodeada de libros
  - Filtrar por título o autor que contenga el texto ingresado (sin distinguir mayúsculas/minúsculas)
- Agregar **tests unitarios** para:
  - Búsqueda exacta
  - Búsqueda parcial
  - Texto vacío

4. Controlador LibroController**:**

- @GetMapping("/buscar") → muestra el formulario
- @PostMapping("/buscar") → devuelve los resultados

💡 **Tips:**

- Validar que no se ejecute búsqueda si el texto está vacío
- Mostrar mensaje si no hay coincidencias
- Inyectar LibroService con constructor
- Usar th:each para renderizar resultados en Thymeleaf
- Empezar por escribir pruebas para el método buscar