Desarrollo de una API REST con Spring Boot y Docker

Objetivo

Desarrollar una aplicación en **Spring Boot** que implemente un CRUD para dos entidades con una relación de **uno a muchos**, aplicando **JPA**, asegurando la seguridad de los endpoints de tipo **POST**, y documentando la API con **Swagger UI**. Además, se deberán realizar **dos pruebas unitarias** a la capa de servicio.

Requisitos

1. **Crear un proyecto en Spring Boot** con las siguientes dependencias:
    - Spring Web
    - Spring Data JPA
    - MySQL
    - Spring Security
    - Spring Boot Starter Test
    - Swagger
2. **Definir dos entidades** con una relación **uno a muchos**.
    - Propongan y ejemplo para sus propias entidades.
    - La entidad principal debe contener dos atributos de tipo LocalDateTime: **fechaInicio** y **fechaFin**.
3. **Implementar los siguientes componentes**:
    - **Repositorio (Repository)** para cada entidad usando JpaRepository.
    - **Servicio (Service)** con la lógica de negocio.
    - **Controlador (Controller)** con los endpoints para CRUD de ambas entidades.
4. **Seguridad**:
    - Proteger todos los endpoints de tipo **POST** con autenticación utilizando **Spring Security**. Usuario: drax182 y Password: x845fg4 y guardenla como variables de sistemas.
5. **Documentación con Swagger UI**:
    - Configurar y documentar los endpoints utilizando **Springdoc OpenAPI**.
    - Asegurarse de que la documentación sea accesible en http://localhost:8080/documentacion
6. **Pruebas Unitarias**:
    - Crear **dos pruebas unitarias** para la capa de servicio usando **JUnit y Mockito**.
    - Probar al menos 4 funcionalidades de cada entidad. ListarTodo, filtrar por fecha de inicio y final mediante queryString. Crear y actualizar.

Entrega

- Subir el código a un contenedor Docker y que funcione.
- Incluir un **README** con instrucciones para ejecutar el proyecto.
- Asegurar que los endpoints POST están protegidos y que Swagger UI está funcionando correctamente.

Criterios de Evaluación

✅ Uso correcto de **Spring Boot y JPA**.
✅ Implementación de la **relación uno a muchos**.
✅ Seguridad aplicada correctamente en los **endpoints POST**.
✅ **Documentación con Swagger UI** bien detallada.
✅ **Pruebas unitarias** funcionales.
✅ Código limpio y estructurado siguiendo **buenas prácticas**.

Extras (Opcionales para mayor puntaje)

⭐ Manejo de excepciones personalizadas como un DTO propio para errores con el mensaje del error y el estatus.

¡Éxito en el desarrollo! 🚀