**Ejercicio 1 de Java Spring: API REST con solo Método GET**

En este ejercicio, se espera se implemente un controlador en una aplicación Java Spring que gestione una API REST. La API debe proporcionar un método GET para recuperar objetos por su ID. Además, los objetos deben cargarse en memoria al arrancar la aplicación.

Requisitos que debes de tener en cuenta:

Crea una clase de modelo llamada Producto con los campos necesarios que se adjuntan en la imagen.

Configura un controlador de Spring llamado ProductController con un método GET que acepte un parámetro de ID y devuelva el objeto correspondiente si existe, o un mensaje de error si no se encuentra.

Crea las clases y métodos necesarias o consideres oportuno para que la aplicación funcione como una aplicación Spring Boot, pero que toda la lógica y el funcionamiento debe estar en el controller.

Crea una clase principal llamada MainApplication que inicie la aplicación Spring Boot.

Ejecuta la aplicación y prueba el endpoint GET para asegurarte de que los objetos se carguen en memoria correctamente y se puedan recuperar por su ID, para esto puedes utilizar una herramienta como Postman.

Además, todo debe estar testeado utilizando la librería de mockito, es muy importante este punto.

Puedes probar a implementar el test primero y luego desarrollar el código (TDD), pero esto es opcional en este punto, lo más importante es que estén los test.

**Ejercicio 2 de Java Spring: API REST Refactorización de servicio**

En este ejercicio, se te solicita refactorizar el controlador de una aplicación Java Spring que gestiona una API REST. Debes mover toda la lógica de negocio relacionada con la recuperación de objetos a un servicio separado. Utiliza el enunciado y el código existente del ejercicio anterior como base.

Requisitos que debes de tener en cuenta:

Crea una clase de servicio llamada ProductService que contenga métodos para manejar la lógica de negocio relacionada con la recuperación de objetos por su ID.

Mueve la lógica de recuperación de objetos del controlador ProcuctController al servicio ProductService. Asegúrate de que el controlador ahora llame al servicio para obtener los objetos por su ID.

Actualiza el controlador y el servicio con las anotaciones y la configuración necesarias para que Spring maneje la inyección de dependencias correctamente.

Verifica que la aplicación funcione correctamente después de la refactorización. Haz las pruebas que consideres oportunas para asegurarte de que la funcionalidad de recuperación de datos funciona correctamente (Postman te puede ayudar a realizar dichas pruebas).

Además, ten en cuenta la legibilidad del código y aplica en la medida de los posible Clean Code y principios SOLID.

Insistimos en esto, puedes probar a implementar el test primero y luego desarrollar el código (TDD), pero esto es opcional, lo más importante es que estén los test, sí o sí.

**Ejercicio 3 de Java Spring: API REST CRUD Productos**

Para este ejercicio, se pide extender la funcionalidad de la aplicación Java Spring existente para permitir operaciones CRUD en los objetos que se cargan en memoria al arrancar la aplicación. El método de lectora ya está desarrollado, es el GET, y por tanto faltaría desarrollar los otros tres: crear, actualizar y borrar. Seguiremos utilizaremos el trabajo realizado hasta ahora como base.

Requisitos que debes de tener en cuenta:

Actualiza el servicio ProductService para incluir métodos que permitan realizar operaciones CRUD en los objetos. Esto incluye métodos para crear, actualizar y borrar objetos en la lista almacenada en memoria.

Modifica el controlador ProductController para incluir endpoints adicionales que permitan realizar las operaciones CRUD en los objetos. Una buena manera de abordarlo podría ser utilizar los métodos del servicio ProductService para manejar estas operaciones.

Al tener que realizar cambios en los datos, crea una gestión de errores para comprobar que la estructura de datos se respeta y que no hay ningún error antes de modificar los valores del objeto a modificar o eliminar, o en caso de crear uno nuevo, que los datos que se reciben cumplen también con esos requisitos.

Añade también la gestión de errores para el endpoint de obtención de productos.

Realiza las pruebas pertinentes con POSTMAN y saca capturas de dichas pruebas para evidenciar que el comportamiento es correcto.

Insistimos en esto, puedes probar a implementar el test primero y luego desarrollar el código (TDD), pero esto es opcional, lo más importante es que estén los test, sí o sí.

**Ejercicio 4 de REST API Creación y conexión BBDD**  
Principio del formulario   
  
Siguiendo las secciones anteriores, ahora debe realizar cambios para que la aplicación existente utilice una base de datos con JPA   
(API de persistencia de Java) en lugar de cargar objetos en la memoria al inicio. Además, debe crear una capa de repositorio para   
administrar las conexiones y la carga de datos desde la base de datos.   
  
Requisitos a tener en cuenta:   
  
Configure la aplicación para que utilice una base de datos compatible con JPA, como MySQL, PostgreSQL o H2 Database.   
Asegúrese de incluir las dependencias necesarias.   
  
Realizar los cambios necesarios para que, además de tener un registro de los productos, también exista un listado de tipos y   
stock asociados a esos productos.   
  
Defina una entidad JPA para representar los objetos de la base de datos. Anote la clase con las anotaciones JPA adecuadas,   
como @Entity, @Id, etc.   
  
Implemente un repositorio utilizando Spring Data JPA. Este repositorio debe ampliar la interfaz JpaRepository y proporcionar métodos   
para realizar operaciones CRUD en la base de datos para la entidad definida en el paso anterior.   
  
Actualice ProductService para usar el repositorio en lugar de los datos en memoria para acceder a los datos de la base de datos.   
Utilice una herramienta de documentación para consultar todos los puntos de conexión y contratos necesarios para utilizar la API.   
Compruebe que la aplicación sigue funcionando correctamente después de la migración y la refactorización.   
Realice las pruebas pertinentes para asegurarse de que los datos se recuperan correctamente de la base de datos y tome capturas de   
pantalla para demostrar estas pruebas y resultados.   
Hacemos hincapié en esto, puede intentar implementar la prueba primero y luego desarrollar el código (TDD), pero esto es opcional.   
Lo más importante es que las pruebas estén ahí, sin falta. Principio del formulario

**Ejercicio 5 de Java Spring: Securización de Endpoints con JWT y Servicio de Login**

Siguiendo los apartados anteriores, agregar seguridad a los endpoints existentes utilizando JWT

(JSON Web Tokens).

Además, debes implementar un nuevo servicio de login para la obtención de dicho token.

Requisitos que debes de tener en cuenta:

Implementa un servicio de login que acepte credenciales de usuario

(por ejemplo, nombre de usuario y contraseña) y devuelva un token JWT válido.

Utiliza una biblioteca como Spring Security para agregar seguridad basada en token a tus endpoints existentes.

Configura Spring Security para requerir un token JWT válido para acceder a los endpoints.

Añade un filtro de seguridad en la configuración de Spring Security para validar el token JWT en las solicitudes entrantes.

Modifica el controlador y los servicios existentes según sea necesario para trabajar con la seguridad implementada.

Asegúrate de que los endpoints estén correctamente protegidos y solo sean accesibles con un token JWT válido.

Actualiza la documentación de la API para incluir detalles sobre cómo obtener y utilizar el token JWT para acceder a los endpoints protegidos.

Implementa dos roles de usuario diferentes: administrador y usuario.

Los roles usuario solo tendrán permisos para realizar peticiones de consulta, y el rol de admin podrá hacer peticiones de consulta, y además,

podrá realizar peticiones para crear, modificar y eliminar.

Ejecuta pruebas para verificar que los endpoints estén protegidos correctamente y que el servicio de login funcione como se espera,

recuerda realizar capturas para evidenciar dichos resultados.

Insistimos en esto, puedes probar a implementar el test primero y luego desarrollar el código (TDD), pero esto es opcional,

lo más importante es que estén los test, sí o sí.