**Ejercicio Practico GraphQL Java & Spring Boot**

Supongamos que queremos crear un sistema de gestión de libros. Para ello, vamos a crear una API GraphQL que nos permita hacer consultas y modificaciones sobre una base de datos de libros. Para simplificar, solo trabajaremos con dos entidades: Libro y Autor.

1. Creación del proyecto

En primer lugar, vamos a crear un proyecto de Spring Boot con Maven. Podemos hacerlo usando alguna herramienta como Spring Initializr o crearlo manualmente.

1. Definición de entidades

A continuación, definiremos las entidades que vamos a utilizar en nuestra aplicación. En este caso, solo tendremos dos: Libro y Autor.

La entidad Libro tiene los siguientes atributos:

* id (String)
* título (String)
* descripción (String)
* fecha de publicación (String)
* autor (Autor)

La entidad Autor tiene los siguientes atributos:

* id (String)
* nombre (String)
* apellido (String)
* fecha de nacimiento (String)

Podemos definir estas entidades como clases Java con sus respectivos atributos, constructores y métodos getters y setters.

1. Definición del repositorio

A continuación, vamos a crear un repositorio que nos permita interactuar con la base de datos. Para ello, crearemos una interfaz que extienda de JpaRepository y que tenga métodos para buscar, crear, actualizar y eliminar libros y autores.

1. Definición de los Data Fetchers

Una vez que tenemos el repositorio, vamos a crear los Data Fetchers. Estos son los objetos que se encargan de obtener los datos de la base de datos y devolverlos en formato GraphQL.

Para ello, crearemos dos clases: BookDataFetcher y AuthorDataFetcher. Cada una de ellas tendrá métodos que se encarguen de obtener los datos de los libros y autores respectivamente.

1. Definición del esquema

Una vez que tenemos los Data Fetchers, podemos definir el esquema GraphQL. Para ello, crearemos un archivo .graphql en el que definiremos los tipos, las consultas y las mutaciones que permitirán interactuar con nuestra API.

1. Creación del servicio GraphQL

Por último, crearemos el servicio GraphQL que se encargará de recibir las consultas y mutaciones y devolver las respuestas adecuadas. Para ello, crearemos una clase GraphQLService que tenga los Data Fetchers y el esquema definidos anteriormente.

En esta clase, utilizaremos la librería graphql-java para generar el objeto GraphQL a partir del esquema y los Data Fetchers.

1. Pruebas

Para probar nuestra API, podemos utilizar algún cliente GraphQL, como GraphiQL, Postman o Insomnia. Con él, podremos enviar consultas y mutaciones a nuestra API y ver las respuestas devueltas.

**Buena Suerte ...**