**anotacion @JsonAlias**

Aprendimos que los nombres de los campos enviados en JSON a la API deben ser idénticos a los nombres de los atributos de las clases DTO, ya que de esta manera Spring puede completar correctamente la información recibida.

Sin embargo, puede ocurrir que un campo se envíe en JSON con un nombre diferente al atributo definido en la clase DTO. Por ejemplo, imagine que se envíe el siguiente JSON a la API:

{

"producto\_id": 12,

"fecha\_compra": "01/01/2022"

}

Y la clase DTO creada para recibir esta información se define de la siguiente manera:

public record DatosCompra(

Long idProducto,

LocalDate fechaCompra

){}

Si esto ocurre, tendremos problemas porque Spring instanciará un objeto del tipo DatosCompra, pero sus atributos no se completarán y quedarán como null debido a que sus nombres son diferentes a los nombres de los campos recibidos en JSON.

Tenemos dos posibles soluciones para esta situación:

1. Renombrar los atributos en el DTO para que tengan el mismo nombre que los campos en JSON;
2. Solicitar que la aplicación cliente que envía las solicitudes a la API cambie los nombres de los campos en el JSON enviado.

La primera opción anteriormente mencionada no es recomendable, ya que los nombres de los campos en JSON no están de acuerdo con el estándar de nomenclatura de atributos utilizado en el lenguaje Java.

La segunda opción sería la más indicada, pero no siempre será posible "obligar" a los clientes de la API a cambiar el estándar de nomenclatura utilizado en los nombres de los campos en JSON.

Para esta situación, existe una tercera opción en la que ninguno de los lados (cliente y API) necesita cambiar los nombres de los campos/atributos. Para ello, solo es necesario utilizar la anotación @JsonAlias:

public record DatosCompra(

@JsonAlias("producto\_id") Long idProducto,

@JsonAlias("fecha\_compra") LocalDate fechaCompra

){}

La anotación @JsonAlias sirve para mapear "alias" alternativos para los campos que se recibirán del JSON, y es posible asignar múltiples alias:

public record DatosCompra(

@JsonAlias({"producto\_id", "id\_producto"}) Long idProducto,

@JsonAlias({"fecha\_compra", "fecha"}) LocalDate fechaCompra

){}

De esta manera, se resuelve el problema, ya que Spring, al recibir el JSON en la solicitud, buscará los campos considerando todos los alias declarados en la anotación @JsonAlias.