✅ PASO A PASO de como aplicar HATEOAS.

✅ PASO 1: Agregar la dependencia de Spring HATEOAS al `pom.xml`

Debes agregar esta dependencia dentro de `<dependencies>`:

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

🔧 **¿Dónde?** Justo después de tus otras dependencias como `spring-boot-starter-web`, `data-jpa`, etc.

✅ PASO 2: Modificar tu modelo ProductoDto para heredar de RepresentationModel

Busca tu archivo ProductoDto.java y haz lo siguiente:

🔧 CAMBIO:

Agrega esta importación:

```
import org.springframework.hateoas.RepresentationModel;
```

Y cambia la declaración de la clase así:

```
package com.productos.dto;
import org.springframework.hateoas.RepresentationModel;
import lombok.*;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ProductoDTO extends RepresentationModel<ProductoDTO> {
    private Integer id;
    private String nombre;
    private String descripcion;
    private Double precioUnitario;
    private String categoria;
    private Boolean activo;
}
```

✅ PASO 3: Agregar enlaces HATEOAS en tu controlador existente

Abre tu clase `ProductoController.java` y modifica el endpoint que retorna un producto específico.

Agrega esta importación:

```
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.linkTo;
import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.methodOn;

//METODO HATEOAS para buscar por ID
@GetMapping("/hateoas/{id}")
public ProductoDTO obtenerHATEOAS(@PathVariable Integer id) {
	ProductoDTO dto = service.obtenerPorId(id);
	dto.add(linkTo(methodOn(ProductoController.class).obtenerHATEOAS(id)).withSelfRel());

	dto.add(linkTo(methodOn(ProductoController.class).obtenerTodosHATEOAS()).withRel("todos"));

	dto.add(linkTo(methodOn(ProductoController.class).eliminar(id)).withRel("eliminar"));

	return dto;
}

//METODO HATEOAS para listar todos los productos utilizando HATEOAS
@GetMapping("/hateoas")
public List<ProductoDTO> obtenerTodosHATEOAS() {
	List<ProductoDTO> lista = service.listar();

	for (ProductoDTO dto : lista) {
		dto.add(linkTo(methodOn(ProductoController.class).obtenerHATEOAS(dto.getId())).withSelfRel());
	}

	return lista;
}
```

🧩 MÉTODOS CLAVE USADOS

🧩 CLASE `ProductoDTO`

```
public class ProductoDTO extends RepresentationModel<ProductoDTO>
```

✅ Esto hace que tu DTO herede la capacidad de contener enlaces HATEOAS en el campo especial `_links`.
Ya no necesitas usar `EntityModel` en el controlador porque los enlaces los puedes agregar directamente al objeto.

✔️ ¿Qué es `RepresentationModel<T>`?

Es una clase base proporcionada por Spring HATEOAS que habilita el soporte para links HATEOAS.
Al heredar de ella, `ProductoDTO` puede usar `.add(Link link)` para incluir enlaces dentro de sí mismo.

✅ MÉTODO: obtenerHATEOAS(Integer id)

```
//METODO HATEOAS para buscar por ID
@GetMapping("/hateoas/{id}")
public ProductoDTO obtenerHATEOAS(@PathVariable Integer id) {
        ProductoDTO dto = service.obtenerPorId(id);
        dto.add(linkTo(methodOn(ProductoController.class).obtenerHATEOAS(id)).withSelfRel());

        dto.add(linkTo(methodOn(ProductoController.class).obtenerTodosHATEOAS()).withRel("todos"));

        dto.add(linkTo(methodOn(ProductoController.class).eliminar(id)).withRel("eliminar"));

        return dto;
}
```

🔎 **Paso a paso:**

1. `ProductoDTO dto = service.obtenerPorId(id);`
   Obtienes el producto desde la capa de servicio.

2. `dto.add(linkTo(...).withSelfRel());`
   Agregas el enlace HATEOAS a sí mismo (el recurso actual).

3. `dto.add(linkTo(...).withRel("todos"));`
   Agrega un link para ver todos los productos.

4. `dto.add(linkTo(...).withRel("eliminar"));`
   Agrega un link que apunta a la operación de eliminación de este producto.

📦 **Resultado esperado:**

json Copiar Editar

```
{
  "id": 10,
  "nombre": "Teclado",
  "descripcion": "Teclado mecánico",
  "precioUnitario": 35.0,
  "categoria": "Periféricos",
  "activo": true,
  "_links": {
    "self": { "href": "http://localhost:8080/productos/hateoas/10" },
    "todos": { "href": "http://localhost:8080/productos/hateoas" },
    "eliminar": { "href": "http://localhost:8080/productos/10" }
  }
}
```

✅ MÉTODO: obtenerTodosHATEOAS()

```
//METODO HATEOAS para listar todos los productos utilizando HATEOAS
@GetMapping("/hateoas")
public List<ProductoDTO> obtenerTodosHATEOAS() {
      List<ProductoDTO> lista = service.listar();

      for (ProductoDTO dto : lista) {
            dto.add(linkTo(methodOn(ProductoController.class).obtenerHATEOAS(dto.getId())).withSelfRel());
      }

      return lista;
}
```

🔎 **Paso a paso:**

1. `List<ProductoDTO> lista = service.listar();`
   Recuperas todos los productos de la base de datos.

2. `for (...) { dto.add(...) }`
   A cada producto le agregas un enlace a sí mismo, usando el método `obtenerHATEOAS(id)`.

📦 **Resultado esperado (para cada producto):**

json — Copiar — Editar

```
{
  "id": 1,
  "nombre": "Monitor",
  ...
  "_links": {
    "self": { "href": "http://localhost:8080/productos/hateoas/1" }
  }
}
```

🔁 RELACIONES (`rel`) USADAS

Relación	Significado
`self`	Link al recurso actual
`"todos"`	Link para ir al listado completo
`"eliminar"`	Link para eliminar este producto

✅ ¿Qué hiciste bien?

✔️ Usaste correctamente el patrón de extender `RepresentationModel` en tu DTO
✔️ Agregaste enlaces de tipo `self`, `rel`, y los estructuraste según buenas prácticas REST
✔️ Separaste los endpoints `normales` de los `HATEOAS`, permitiendo que el cliente elija