**Retorno de JSON** nas exceções

**🔍 1. As anotações**

@ControllerAdvice

@RestController

Essa combinação indica que:

* @ControllerAdvice → intercepta exceções lançadas por qualquer controller da aplicação.
* @RestController → faz com que o retorno de todos os métodos seja convertido automaticamente em **JSON**, sem precisar usar @ResponseBody em cada um.

💡 Você poderia inclusive substituir por:

@RestControllerAdvice

que é exatamente o mesmo efeito (é uma junção das duas anotações).

**⚙️ 2. O método @ExceptionHandler**

Cada método anotado com @ExceptionHandler intercepta um tipo de exceção específico.  
Por exemplo:

@ExceptionHandler(ResourceNotFoundException.class)

significa: “se um controller lançar ResourceNotFoundException, chame este método”.

**🧱 3. O retorno ResponseEntity<ExceptionResponse>**

Você retorna um objeto ResponseEntity, que contém:

* O corpo da resposta (ExceptionResponse),
* E o status HTTP (HttpStatus.NOT\_FOUND, HttpStatus.INTERNAL\_SERVER\_ERROR, etc.).

Como o tipo genérico é um **objeto Java (ExceptionResponse)**, o Spring usa automaticamente o **Jackson** para converter isso em **JSON**.

**🧾 4. Resultado prático (exemplo de resposta JSON)**

Se o seu ExceptionResponse tiver algo assim:

public class ExceptionResponse {

private Date timestamp;

private String message;

private String details;

// construtor, getters, setters...

}

Então, uma exceção lançada geraria uma resposta JSON mais ou menos assim:

{

"timestamp": "2025-10-27T16:45:12.345+00:00",

"message": "Resource not found",

"details": "uri=/api/users/999"

}

✅ **Resumindo:**  
Sim, o @ExceptionHandler aqui **retorna um JSON automaticamente**, graças ao @RestController (ou @RestControllerAdvice) e ao uso do ResponseEntity<ExceptionResponse>.