**✅ Root Cause Analysis — Problema com paginação e serialização JSON no Spring Boot**

**🔍 Causa Raiz 1 – Jackson + @JsonView**

O endpoint:

@GetMapping(produces = MediaType.APPLICATION\_JSON\_VALUE)

@JsonView(UsuarioView.Listagem.class)

public Page<UsuarioDto> listarPaginado(...)

retornava uma resposta JSON vazia nos objetos da lista, como:

{

}

**❗ Diagnóstico Real**

Mesmo com a @JsonView aplicada corretamente no controller e os campos do DTO anotados com a mesma view, **o conteúdo do Page<UsuarioDto> era serializado como {}**.

Isso ocorre por **uma limitação conhecida do Jackson ao lidar com tipos genéricos e PageImpl**:

* O PageImpl é um wrapper genérico.
* O Jackson **não consegue inferir corretamente o tipo real dos elementos dentro de Page<T> quando está serializando com @JsonView**.
* Isso causa a **perda da associação entre a view ativa e os campos internos dos elementos da lista**.
* Resultado: todos os campos anotados com @JsonView são ignorados, mesmo com a view declarada.

📌 Esse bug é conhecido na comunidade e discutido em várias issues do Jackson e do Spring.

**⚠️ Causa Raiz 2 – Alerta do Spring Boot com PageImpl**

Mesmo com a serialização funcionando, apareceu o seguinte aviso:

WARN ... Serializing PageImpl instances as-is is not supported

Esse aviso ocorre porque a classe PageImpl é retornada diretamente e sua estrutura interna (como Pageable, Sort, etc.) pode mudar entre versões. Isso quebra a estabilidade do JSON para clientes externos.

O Spring recomenda:

Usar PagedModel com Spring HATEOAS, ou

Usar DTOs para serializar a resposta de forma estável.

✔️ Correção aplicada: PageDto<T>

Foi criada uma classe customizada PageDto<T>:

public class PageDto<T> {

private List<T> content;

private int pageNumber;

private int pageSize;

private long totalElements;

private int totalPages;

private boolean last;

private boolean first;

private int numberOfElements;

}

E usada da seguinte forma:

*@Getter*

*@Setter*

*@NoArgsConstructor*

public class PageDto<T> {

private List<T> content;

private int pageNumber;

private int pageSize;

private long totalElements;

private int totalPages;

private boolean last;

public PageDto(Page<T> page) {

this.content = page.getContent();

this.pageNumber = page.getNumber();

this.pageSize = page.getSize();

this.totalElements = page.getTotalElements();

this.totalPages = page.getTotalPages();

this.last = page.isLast();

}

// getters

}

Com isso, evitamos:

Dependência do Spring HATEOAS

Problemas de compatibilidade futura

Sobrecarga no JSON gerado

✅ Conclusão e Benefícios

Aspecto Resultado

Serialização com Jackson Corrigida ao evitar @JsonView no DTO retornado

Padrão de Resposta JSON Estabilizado via PageDto<T> customizado

Acoplamento com Spring Reduzido, sem necessidade de usar HATEOAS

Manutenção futura Facilitada por menor dependência e JSON mais claro e enxuto

Performance e legibilidade