Quando se está trabalhando com uma **quantidade grande de dados**, uma boa ideia é **diminuir a carga do que é apresentado através de paginação**. A ideia da **paginação é óbvia**, **apresentar "blocos" de registro invés de todos eles.**

A **paginação** sendo algo **tão comumente utilizado** é lógico que o **Spring Data JPA** já vai ter uma **implementação para isso**, e esse repositório é o **PagingAndSortingRepository** ***(que aliás estende de CRUD repository):***

```
@Repository
public interface FuncionarioRepository extends PagingAndSortingRepository<Funcionario, Long> {

    @Query("from Funcionario f JOIN FETCH f.cargo WHERE f.nome = :nome")
    List<Funcionario> findByNome(String nome);
```

Uma **vez estendida a interface** de **PagingAndSorting** o **repositório vai conseguir trabalhar com objetos do tipo *"pagina"***.

Então agora minhas **consultas** **podem receber um** **Pageable** que pode ser criado com o **número da página *(caso queira ir para uma específica)*, a quantidade de registros por página** e **algum tipo de ordenação.** A **consulta** que receber o **Pageable** deve retornar uma **Página/Page:**

```
private void listarTodos(){
    Pageable paginacao = PageRequest.of( page: 1,  size: 5, Sort.unsorted());

    Page<Funcionario> funcionarios = funcionarioRepository.findAll(paginacao);


    funcionarios.forEach(System.out::println);
}
```

A vantagem de usar um **Page** no **retorno da consulta** é justamente poder **extrair informações uteis dele**, como a *página atual que ele representa, ou o total de elementos da consulta* e etc:

```
Page<Funcionario> funcionarios = funcionarioRepository.findAll(paginacao);

 System.out.println(funcionarios);
 System.out.println("Pagina atual: "+ funcionarios.getNumber());
 System.out.println("Total de funcionários: "+ funcionarios.getTotalElements());
```

Pra **ordenação** é bem simples, dá pra usar um **Sort.directrion pra definir a ordem** e o **atributo do objeto** que queremos passar como critério da ordenação:

```
Pageable paginacao = PageRequest.of(pagina,  size: 5, Sort.by(Sort.Direction.ASC,  ...properties: "nome"));
```

*****Relacionamentos Lazy ainda são meio difíceis de se resolver quando usamos implementações padrões do SDJ para consultas, to buscando respostas para resolver isso sem precisar usar más páticas*****