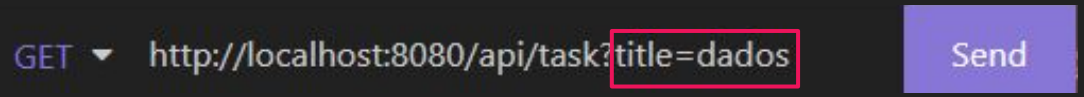
JAVA ADVANCED

#04
BUSCA E PAGINAÇÃO

João Carlos Lima

QUERY PARAMETERS




```
@GetMapping("/api/task")
@ResponseBody
public List<Task> index( @RequestParam String title) {

    return repository.findByTitleLike("%" + title + "%");

}
```

PAGINAÇÃO

Offset

Cursor

page=1 size=5

PAGINAÇÃO

Offset

Cursor

page=1 size=5

PAGINAÇÃO

Offset

Cursor

page=1 size=10

PAGINAÇÃO

Offset

Cursor

page=2 size=5

PAGINAÇÃO

Offset

Cursor

page=3 size=5

PAGINAÇÃO

Offset

Cursor

previous=null

next=4

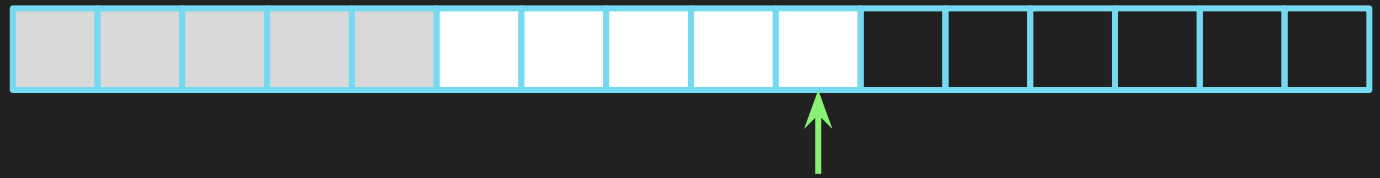
PAGINAÇÃO

Offset

Cursor

previous=4

next=9



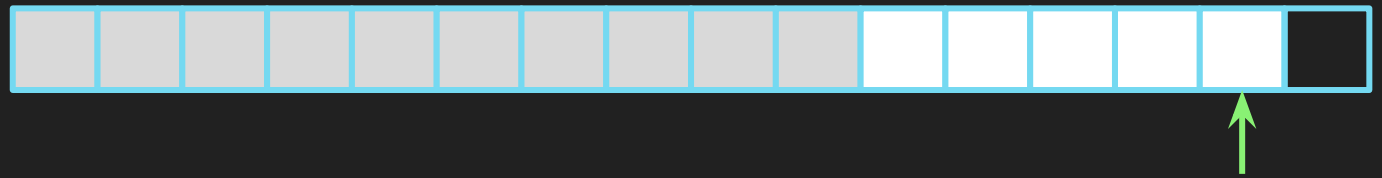
PAGINAÇÃO

Offset

Cursor

previous=9

next=14



PAGINAÇÃO E ORDENAÇÃO

```
@GetMapping
    public List<Despesa> index( @RequestParam(defaultValue = "-1") int pagina, @RequestParam(defaultValue = "0") int tamanho) {

        return repository.findAll(Pageable.ofSize(tamanho).withPage(pagina)).getContent();
    }
```

```
@GetMapping
public Page<Despesa> index(Pageable pageable) {
    return repository.findAll(pageable);
}
```

EXAMPLE E EXAMPLE MATCHER

O Example e o ExampleMatcher são recursos do **Spring Data JPA** que permitem fazer buscas dinâmicas baseadas em uma **instância de exemplo** (um objeto preenchido com os valores que queremos buscar). Isso é útil quando queremos evitar escrever consultas manuais ou usar muitas anotações. O ExampleMatcher serve para configurar **como** a comparação será feita — por exemplo, se a busca deve ignorar maiúsculas/minúsculas, usar *contains* em vez de *equals*, etc.

```java
public List<Personagem> buscarComExample(String nome, String classe) {
    Personagem exemplo = new Personagem();
    exemplo.setNome(nome); // Ex: "ana"
    exemplo.setClasse(classe); // Ex: "mago"

    ExampleMatcher matcher = ExampleMatcher
        .matchingAll()
        .withIgnoreCase()
        .withStringMatcher(ExampleMatcher.StringMatcher.CONTAINING);

    Example<Personagem> filtro = Example.of(exemplo, matcher);

    return personagemRepository.findAll(filtro);
}
```

SPECIFICATION

A interface Specification<T> permite criar **consultas dinâmicas e reutilizáveis** usando a API de Criteria do JPA. Diferente de Example, que compara objetos diretamente, Specification é mais poderosa: ela permite fazer buscas por intervalos, aplicar operadores (>, <, IN, LIKE) e combinar múltiplas condições usando AND ou OR. Ela é ideal quando os filtros precisam ser **mais complexos ou flexíveis** — por exemplo, buscar registros com valores entre limites, ou aplicar múltiplos filtros opcionais vindos de uma DTO.

```java
public class CursoSpecification {

    public static Specification<Curso> filtrar(CursoFilter filtro) {
        return (root, query, cb) -> {
            List<Predicate> predicates = new ArrayList<>();

            if (filtro.getNome() != null) {
                predicates.add(cb.like(cb.lower(root.get("nome")), "%" + filtro.getNome().toLowerCase() + "%"));
            }

            if (filtro.getCategoria() != null) {
                predicates.add(cb.equal(root.get("categoria"), filtro.getCategoria()));
            }

            if (filtro.getCargaHorariaMin() != null) {
                predicates.add(cb.greaterThanOrEqualTo(root.get("cargaHoraria"), filtro.getCargaHorariaMin()));
            }

            if (filtro.getCargaHorariaMax() != null) {
                predicates.add(cb.lessThanOrEqualTo(root.get("cargaHoraria"), filtro.getCargaHorariaMax()));
    }
}
```

CRITÉRIOS DE USO

Critério/Recurso	Query Method	Example Matcher	Specification
Facilidade de uso	Fácil	Médio	Difícil
Consulta baseada em objeto	❌	✅	❌
Busca parcial (contains, like)	✅	✅	✅
Comparação com operadores	❌	❌	✅
Consulta com múltiplos filtros	❌	✅	✅
Filtros opcionais	❌	✅	✅
Reutilização de Filtros	❌	❌	✅
Boa para filtros simples	✅	✅	❌
Boa para filtros complexos	❌	❌	✅