03

Para saber mais: Mais Derived Queries

Em vídeo, vimos como buscar funcionários pelo método `findByNome` dentro do repositório:

```
@Repository
public interface FuncionarioRepository extends
CrudRepository<Funcionario, Integer> {
    List<Funcionario> findByNome(String nome);
}
```

COPIAR CÓDIGO

As queries derivadas são simples, mas poderosas, e oferecem mais variações e recursos. Seguem alguns exemplos:

Usando Like

Para executar um *like* (e não um *equals*, como no exemplo), use:

```
List<Funcionario> findByNomeLike(String nome);
```

COPIAR CÓDIGO

O valor do parâmetro `nome` deve usar o *pattern*, por exemplo:

```
String nome = "%maria%";
```

COPIAR CÓDIGO

Starting e Ending

Você também pode buscar os funcionários pelo prefixo ou sufixo:

```
List<Funcionario> findByNomeEndingWith(String nome)
```

COPIAR CÓDIGO

Ou:

```
List<Funcionario> findByNomeStartingWith(String nome)
```

COPIAR CÓDIGO

Null e not Null

Igualmente podemos pesquisar elemento nulos ou não nulos:

```
List<Funcionario> findByNomeIsNull()
```

COPIAR CÓDIGO

Ou não nulos com:

```
List<Funcionario> findByNomeIsNotNull()
```

COPIAR CÓDIGO

Ordenação

Ainda vamos falar sobre ordenação e páginas, mas claro que a *Derived Query* pode dar suporte:

```
List<Funcionario> findByNomeOrderByNomeAsc(String nome);
```

COPIAR CÓDIGO

Métodos longos

E como dica, como definimos os critérios de pesquisa por meio do nome do método, precisamos ter cuidado para não criar nomes gigantes e prejudicar a legibilidade. Nesse caso devemos favorecer as consultas com JPQL apresentadas no próximo vídeo.

Documentação

Por fim aqui está a documentação do Spring Data JPA (https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods), com mais exemplos.