

Logo da Orange Talents

# Mapeando queries explicitamente no Micronaut Data

## Antes de começar

Para complementar o curso, recomendamos estudar a documentação oficial do Micronaut sobre [Writing Queries](https://micronaut-projects.github.io/micronaut-data/latest/guide/#querying) caso você não tenho estudado ainda. Nela é possível ter uma idéia mais clara do poder e das possibilidades oferecidas pelo framework.

Além disso, sentir-se confortável com JPA e Hibernate será de grande ajuda ao trabalhar com persistência no Micronaut. Não à toa os conhecimentos adquiridos no curso da ALURA de [Persistência com JPA: Introdução ao Hibernate](https://www.alura.com.br/curso-online-persistencia-jpa-introducao-hibernate) podem ser úteis em problemas semelhantes a este.

## Cenário:

Imagine que temos como primeira tarefa em um novo time (squad) escrever uma consulta para um relatório especifico no sistema. Por se tratar de um projeto com Micronaut, banco relacional e JPA/Hibernate, nosso tech lead nos contextualizou com as seguintes entidades da JPA utilizadas atualmente no nosso modelo de domínio e com o repository de notas fiscais:

@Repository  
interface NotaFiscalRepository : JpaRepository<NotaFiscal, Long> {  
  
 // outros métodos implementados pelo time  
}  
  
@Entity  
class NotaFiscal(  
 @Id  
 @GeneratedValue  
 val id: Long,  
 val numero: String,   
 val serie: String,   
 val data: LocalDate  
 @OneToMany(fetch = FetchType.LAZY)  
 val itens: List<ItemDeNota>,  
 // outros atributos  
)  
  
@Entity  
class ItemDeNota(  
 @Id  
 @GeneratedValue  
 val id: Long,  
 val valor: BigDecimal,   
 val quantidade: Int,  
 // outro atributos  
)

A idéia do relatório é bem simples, basicamente temos que carregar uma nota fiscal por id com todos seus itens para então exibir no aplicativo mobile do nosso usuário. Nesse caso, precisamos de fato de todos os campos da nota fiscal e dos itens, porém tem um ponto de atenção importante que o tech lead nos alertou: por se tratar de um sistema com uma volumetria de dados razoável e milhares de usuários ao redor do país, é importante que nossa consulta carregue todos os dados numa única ida e volta (roundtrip) ao banco de dados, dessa forma os itens devem ser carregados juntamente com nota fiscal consultada em um único comando SELECT.

Como você faria para implementar uma consulta para carregar uma nota fiscal por id com todos seu itens utilizando o repository do Micronaut já existente no projeto?

## O que seria bom ver nessa resposta?

* **Peso 5**: Criar um método no repository com uma query explicita passando o id como parâmetro. Aqui tanto faz a query ser escrita em JPQL ou SQL nativo, o importante é que seja via uma das anotações suportadas pelo Micronaut: @Query, @Join ou @EntityGraph;
* **Peso 4**: Garantir que somente uma única query seja disparada pelo Micronaut carregando os dados da nota fiscal juntamente com os dados dos seus itens;
* **Peso 1**: Apresentar algum domínio de JPQL, por exemplo utilizando o recurso join fetch da JPA;

## O que penaliza sua resposta?

* **Penalidade -5**: Alterar o mapeamento do relacionamento entre nota e itens para FetchType.EAGER;

## Resposta do Especialista:

* Crio um novo método na interface do repository e o anoto com a anotação @Query do Micronaut para que eu consiga escrever uma query JPQL customizada de tal forma que carregue a nota fiscal com seus itens:
* @Query("select n from NotaFiscal n where n.id = :id")  
   fun findByIdWithItens(id: Long): NotaFiscal
* Testo o método verificando o SQL gerado pelo Micronaut no console da IDE (assumo que a configuração show\_sql do Hibernate está habilitada). Se houver problemas de sintaxe eu vou corrigindo. Eu aproveito para constatar que somente uma única query foi disparada para o banco de dados para carregar a nota fiscal por id, afinal temos um relacionamento LAZY entre a nota e seus itens (a segunda query para carregar os itens aconteceria apenas ao invocar a property itens da entidade no momento de gerar o relatório);
* Com a JPQL funcionando, agora **eu preciso alterá-la para garantir que tanto a nota quanto seus itens sejam carregados com apenas uma única query**. Para isso eu utilizo o recurso join fetch da JPA que faz com que um relacionamento LAZY se comporte como EAGER, ou seja, que as entidades filhas sejam carregadas juntamente com a entidade pai. No final, a anotação @Query teria uma query parecida com essa:
* select n   
   from NotaFiscal n   
   inner join fetch n.itens -- aqui vai o join fetch  
   where n.id = :id
* Por fim, testo a consulta verificando se o Micronaut gerou um único comando SELECT mas dessa vez com um INNER JOIN explicito entre as tabelas de nota fiscal e de itens carregando os dados de ambas as entidades;