

Para saber mais: Class based Projection

Vimos em vídeo como definir uma projeção baseada na interface:

```
public interface FuncionarioProjecao {  
    Integer getId();  
    String getNome();  
    Double getSalario();  
}
```

Essa forma de projeção também é chamada de **Interface based Projection**.

Como alternativa, podemos também usar uma classe com o mesmo propósito:

```
public class FuncionarioDto {  
    private Integer id;  
    private String nome;  
    private Double salario;  
  
    //getter e setter  
  
    //construtor recebendo os atributos  
    //na ordem da query  
}
```

E no nosso repositório:

```
@Query(value = "SELECT f.id, f.nome, f.salario FROM funcionarios f", nativeQuery = true)  
List<FuncionarioDto> findFuncionarioSalarioComProjecaoClasse();
```

Repare na classe `FuncionarioDto` como tipo genérico da lista no retorno do método.

Uma classe dá muito mais trabalho de escrever e manter, mas pode ter uma vantagem, pois podemos adicionar métodos mais específicos que podem fazer sentido para a view (por exemplo, os de formatação).

obs.: o sufixo **Dto** é muito comum para esse tipo de classe auxiliar, e significa *Data Transfer Object*.