Na maioria das vezes não precisamos de um **SELECT * .** Precisamos apenas de um **SELECT** que busque **colunas específicas** de um determina **entidade.** Isso é bem fácil de se fazer utilizando **JPQL** ou **nativeQuery**, a questão é **"que tipo de objeto deve ser retornado?".**

Ai que entra o recurso de **Projeção** do **Spring Data JPA,** a ideia é criar uma **Interface** de **Projeção** da **Entidade** que irá conter ***métodos get*** dos **atributos da entidade**, a **Interface** serve para encapsular o valor de retorno da consulta dentro desses métodos:

```java
@Entity
@Table(name = "funcionarios")
public class Funcionario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;

    private String cpf;

    private BigDecimal salario;
```

```java
public interface FuncionarioProjecao {

    Long getId();

    String getNome();

    BigDecimal getSalario();
}
```

Uma vez feita a **Interface** de **projeção** é necessário então criar a **consulta no repositório** e fazer ela utilizar a **interface projetada** como **retorno**:

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

    @Query("SELECT f.nome as nome, f.salario as salario, f.id as id FROM Funcionario f")
    List<FuncionarioProjecao> buscaProjetada();
```

*****Consultas em JPQL precisam necessariamente que os campos usem alias(apelido), consultas em query nativa não precisam*****

Depois disso basta usar os **métodos da Interface** para recuperar os **atributos**:

```java
private void consultaProjetada(){
    List<FuncionarioProjecao> funcionario = funcionarioRepository.buscaProjetada();
    funcionario.forEach(f -> System.out.print("| "+f.getNome()+"| "+f.getId()+"| "+f.getSalario()));
    System.out.println();
}
```

```
| Lucas| 1| 2300.00| Jorge matheus| 2| 1670.80| Laulo Loberto| 3| 6600.60| Ozymandias| 4| 3200.60
```