



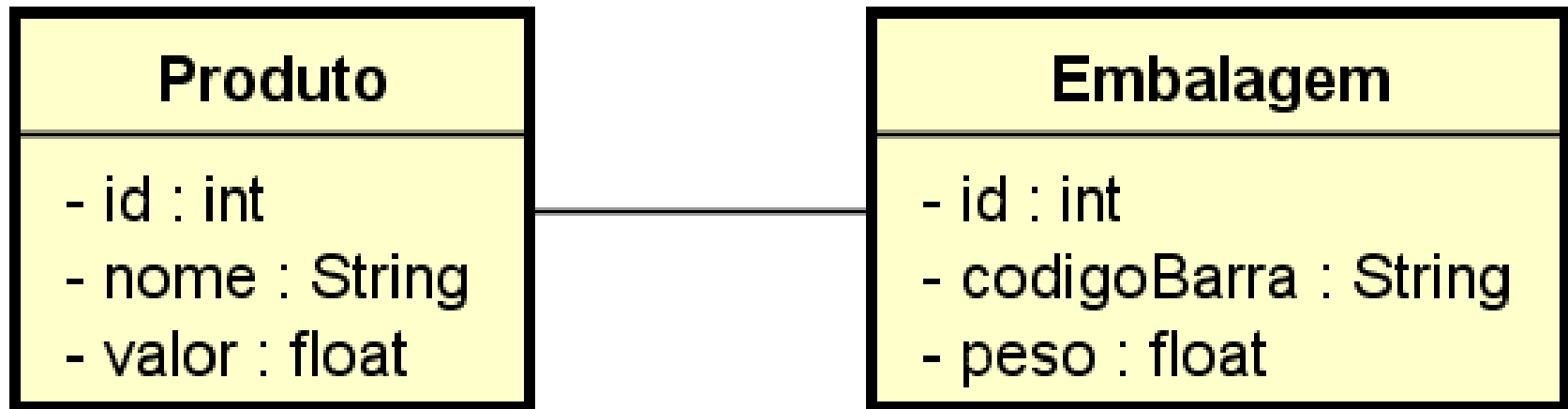
INSTITUTO FEDERAL
Rio Grande do Sul

Desenvolvimento de Sistemas I

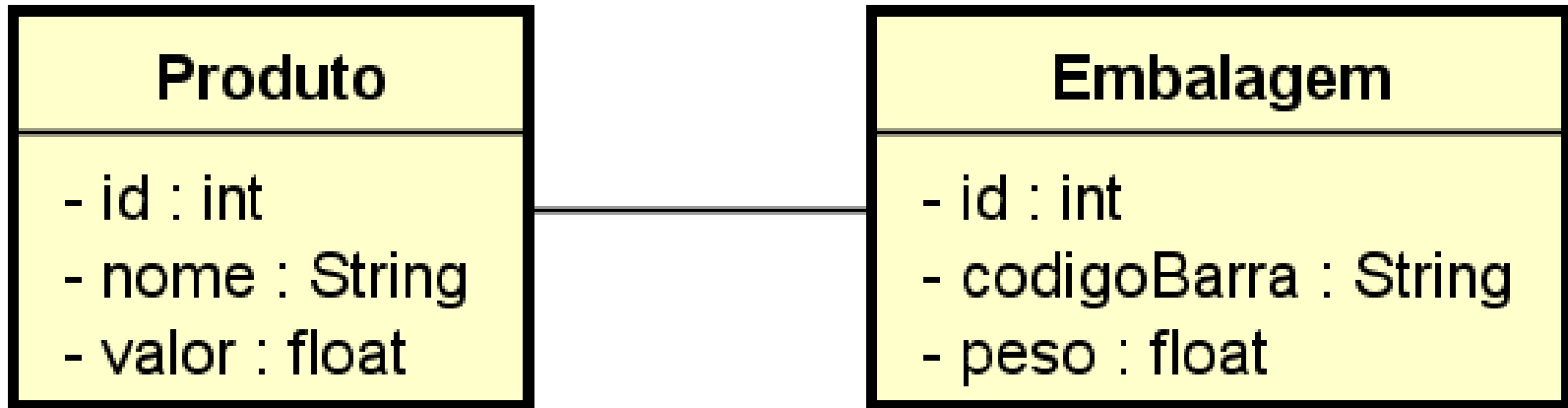
*Associação de classes com JPA e
REST*

Professor: Jezer Machado de Oliveira

Associação entre classes



Associação entre classes



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```

...

...



Associação entre classes

Uma classe A está associada a uma classe B quando possui um atributo cujo tipo é B, e/ou B possui um atributo cujo o tipo é A

```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```

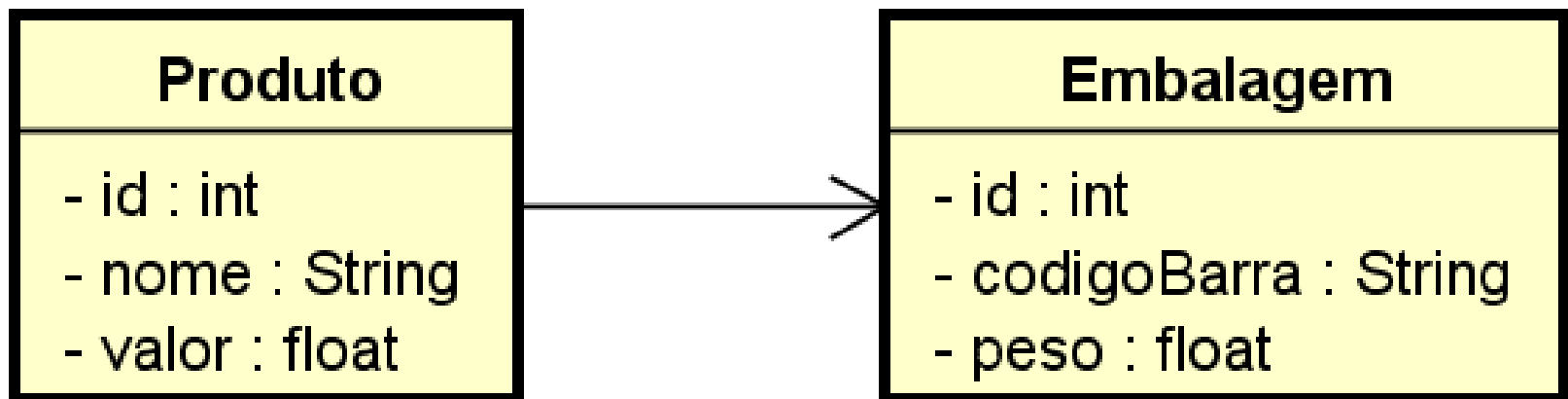
Navegabilidade

- Indica que, a partir de um objeto, é possível chegar a outro que esteja associado a ele

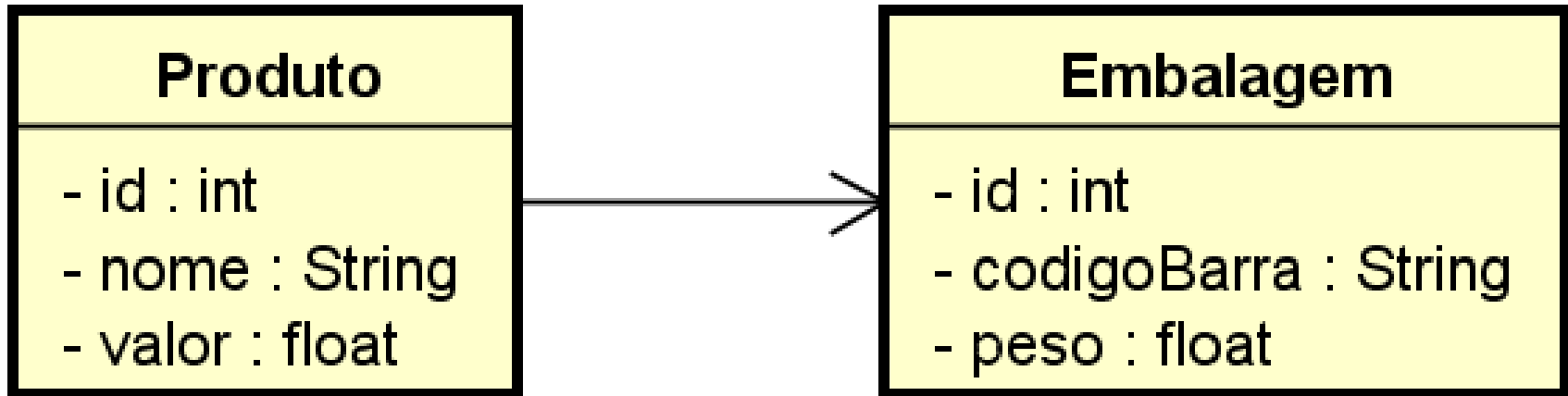


Navegabilidade

- Unidirecional
 - Apenas em um sentido
 - Somente umas das classes guarda a referência da outra



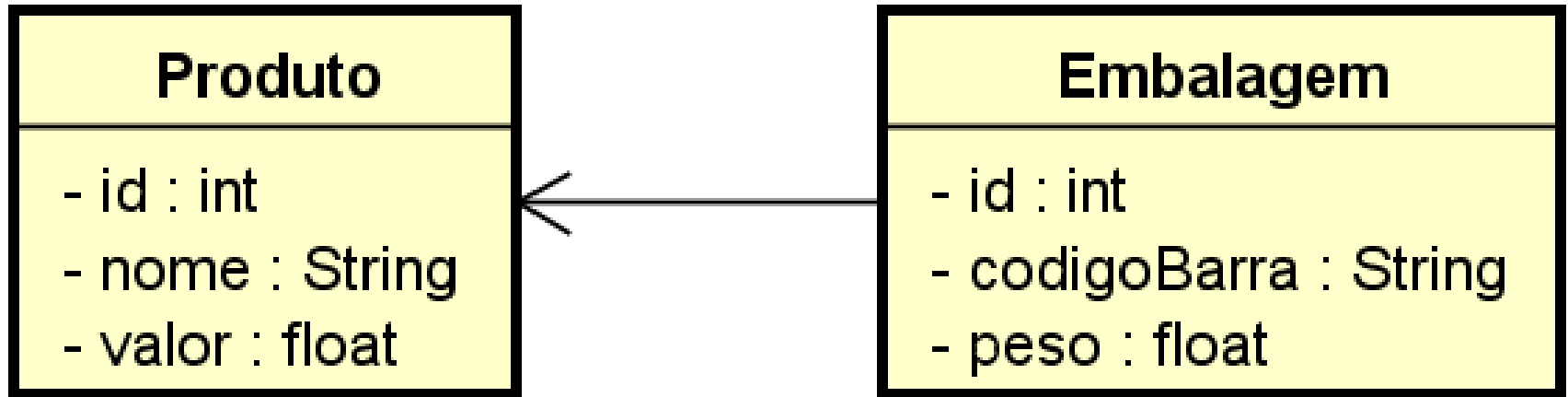
Navegabilidade - Unidirecional



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```

Navegabilidade - Unidirecional

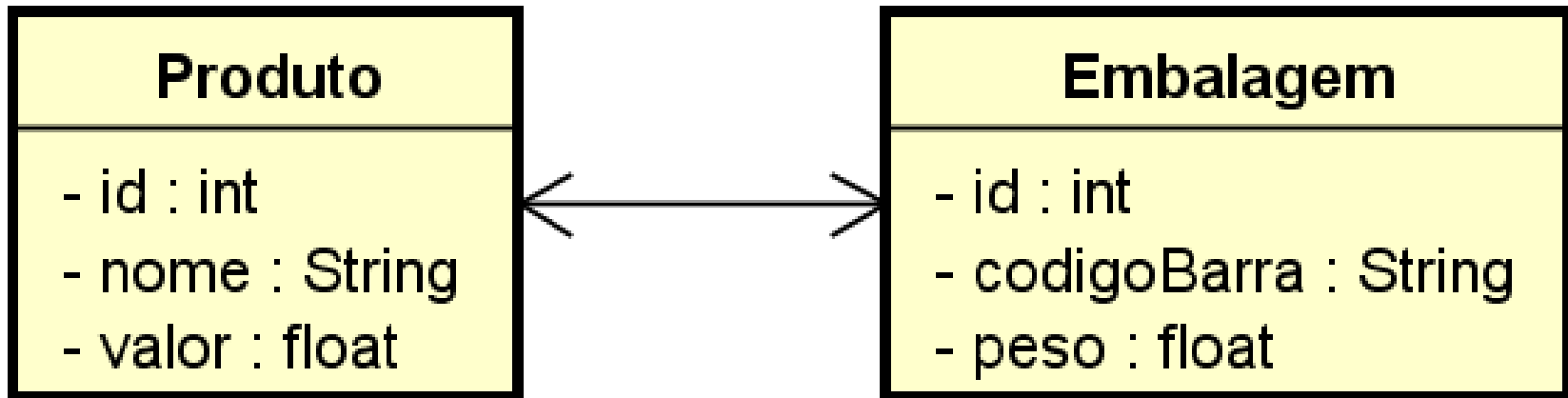


```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

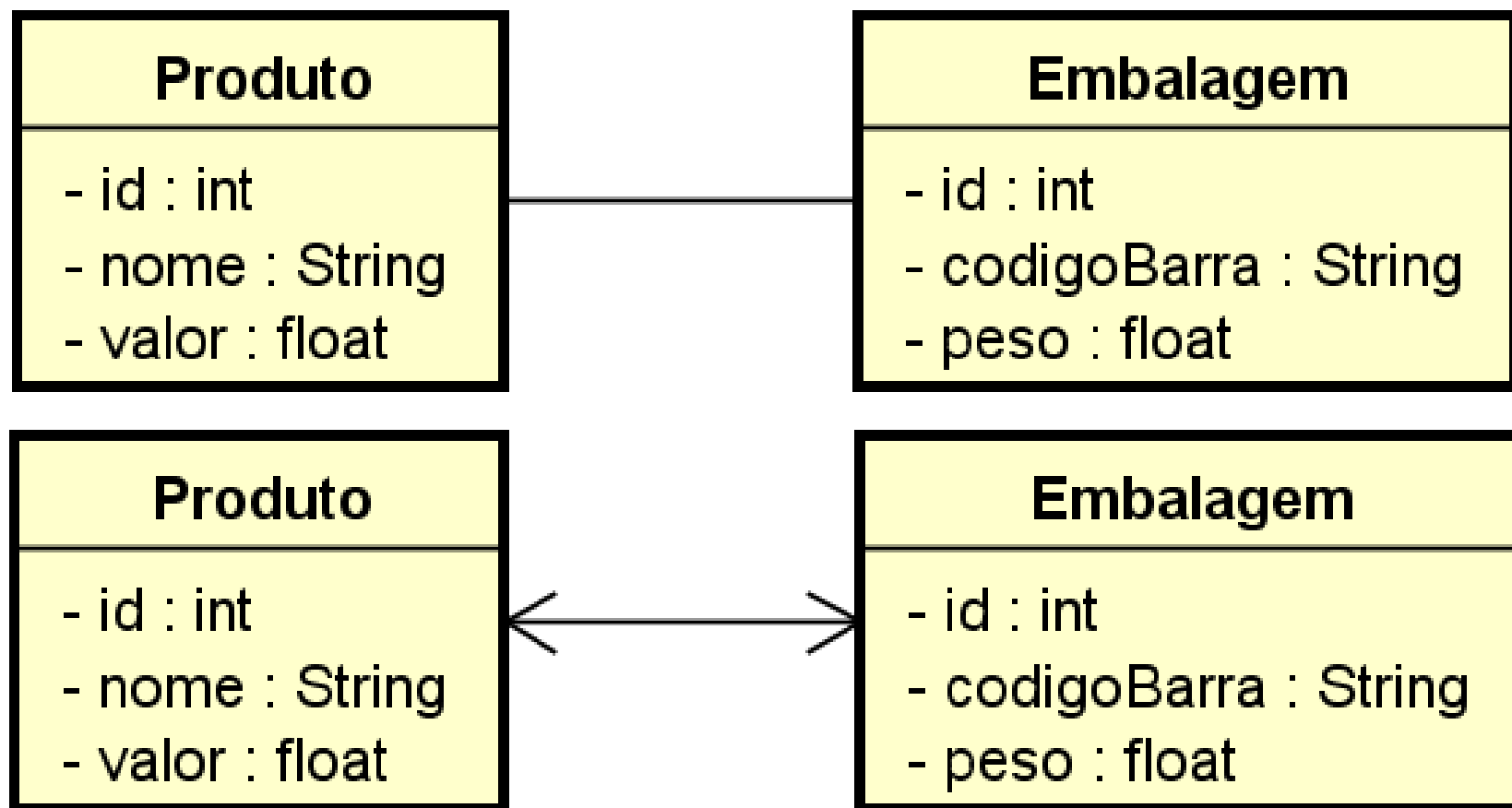
```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```


Navegabilidade - Bidirecional

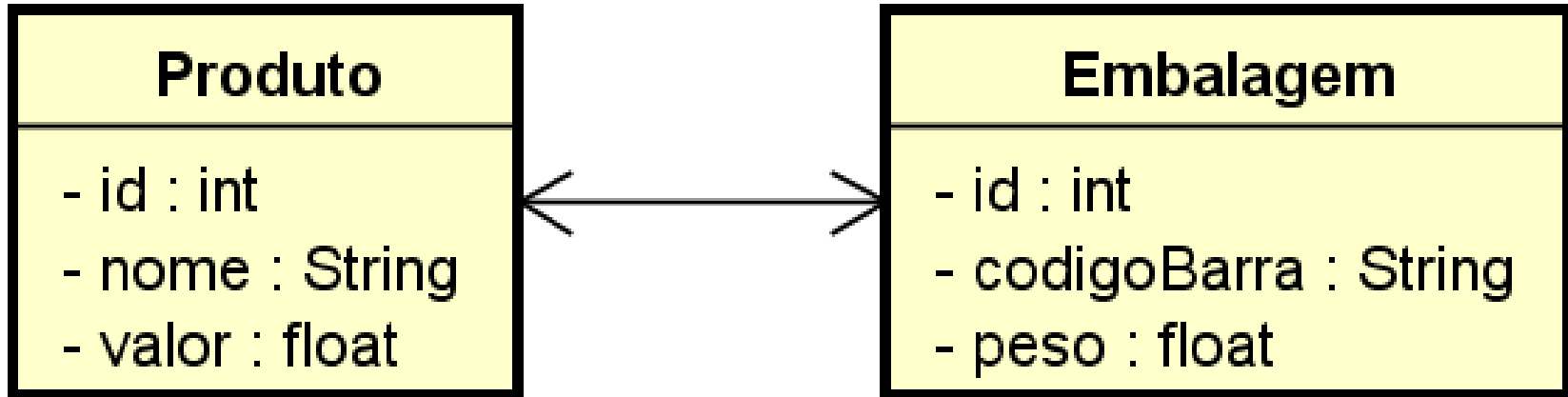
- Bidirecional
 - Sentido da associação é duplo
 - Cada classe guarda a referência de sua contraparte



Navegabilidade - Bidirecional



Navegabilidade - Bidirecional



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```

Multiplicidade

Refere-se a quantos objetos de uma classe, estão relacionados a quantos, de outra classe em uma associação:

- Quantos fornecedores um produto pode ter?
- Quantos produtos um fornecedor pode oferecer?
- Quantas pessoas podem pegar o mesmo livro?
- Quantos livros uma pessoa pode pegar?
- Quantos motores um carro pode ter?
- Em quantos carros um motor pode estar?

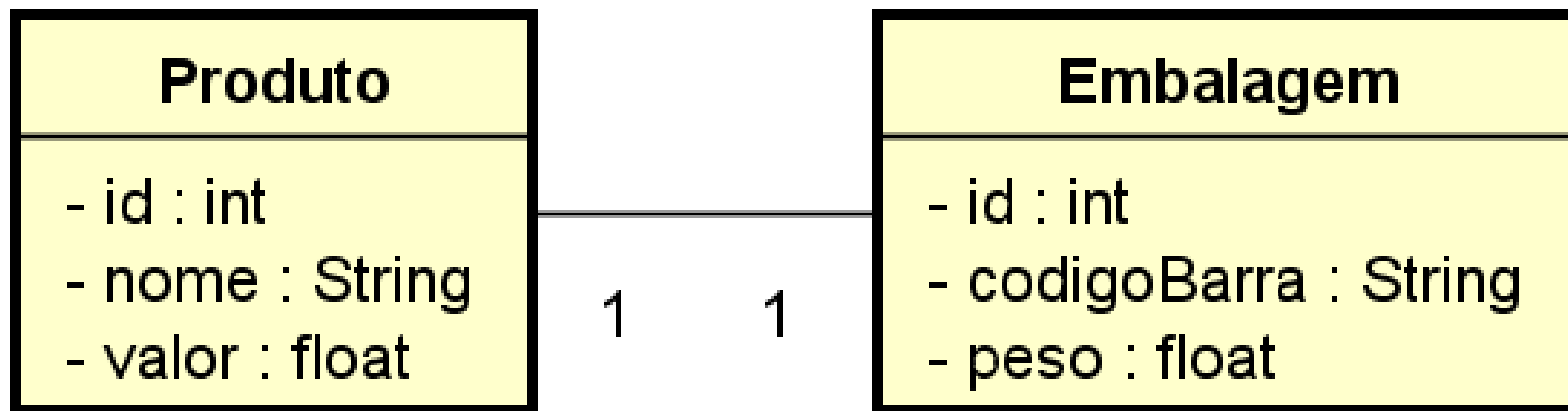


Um para um (one-to-one)

- Cada objeto de uma classe estará associado a, exatamente, um objeto de outra classe
- Normalmente associado a composição
- Exemplos
 - Um carro possui apenas um motor
 - Um empregado possui um curriculum
 - Um usuário possui um login



Um para um (one-to-one)



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private Embalagem embalagem;  
    ...  
}
```

```
public class Embalagem {  
    private int id;  
    private String codigoBarra;  
    private float peso;  
    private Produto produto;  
    ...  
}
```

...

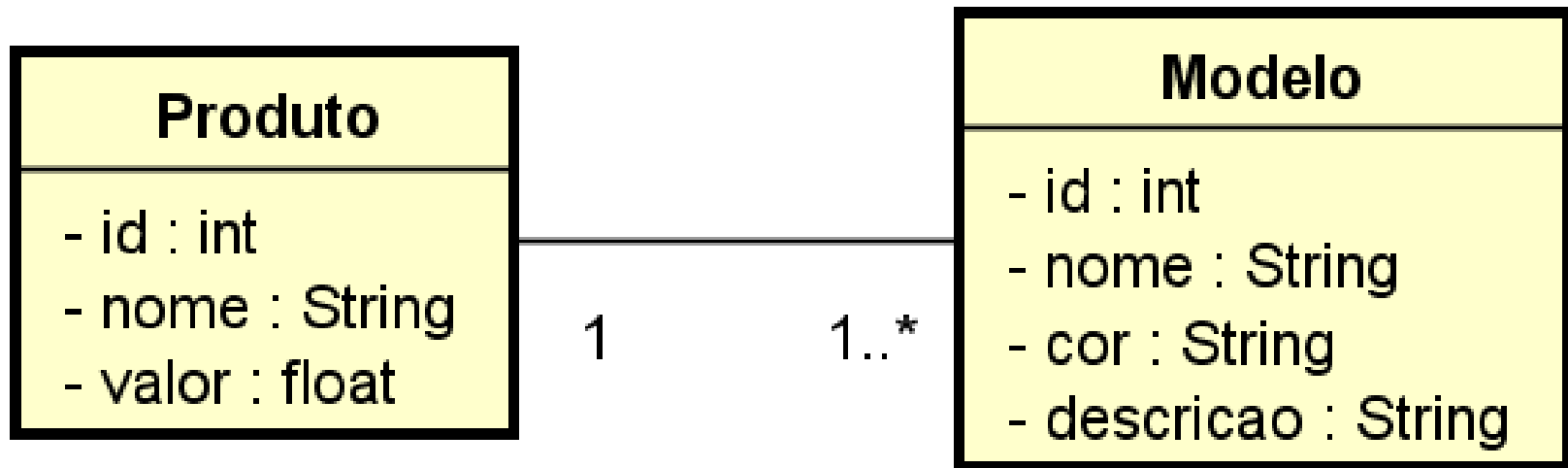
...

Um para muitos ou muitos para um (one-to-many/many-to-one)

- Cada objeto de uma classe pode estar associado a muitos objetos de outra classe
- Questão de perspectiva
- Exemplos
 - Um carro possui vários bancos, mas um banco só pode estar em um carro
 - Uma empresa possui vários empregados, mas um empregado só pode trabalhar em uma empresa



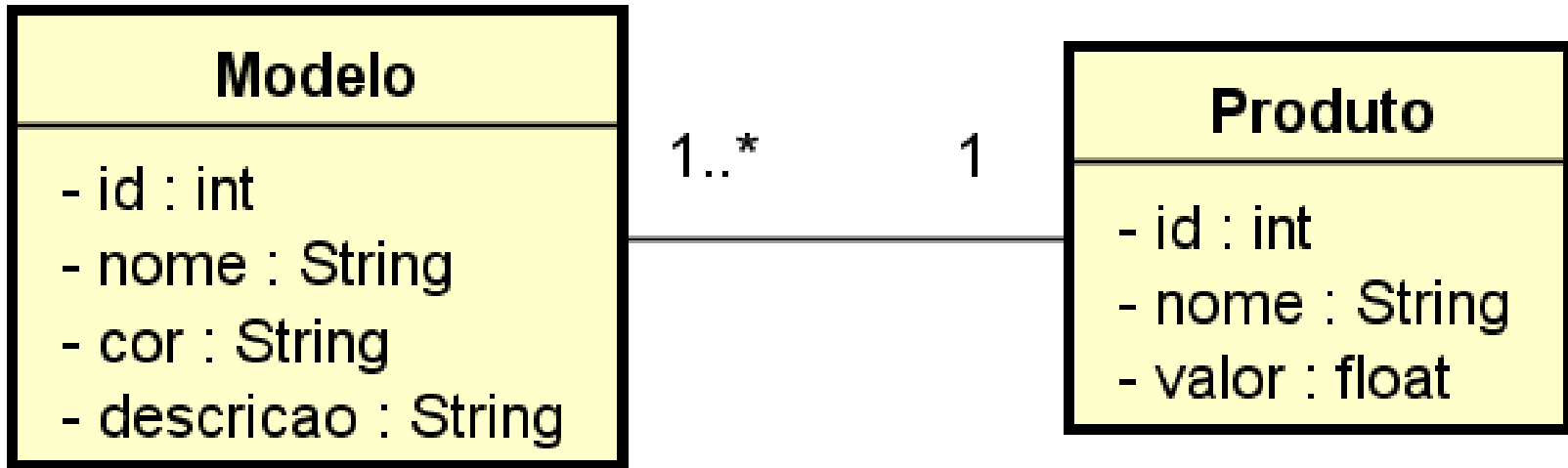
Um para muitos (one-to-many)



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private List<Modelo> modelos;  
    ...  
}
```

```
public class Modelo {  
    private int id;  
    private String nome;  
    private String cor;  
    private String descricao;  
    private Produto produto;  
    ...  
}
```


Muitos para um (many-to-one)



```
public class Modelo {  
    private int id;  
    private String nome;  
    private String cor;  
    private String descricao;  
    private Produto produto;
```

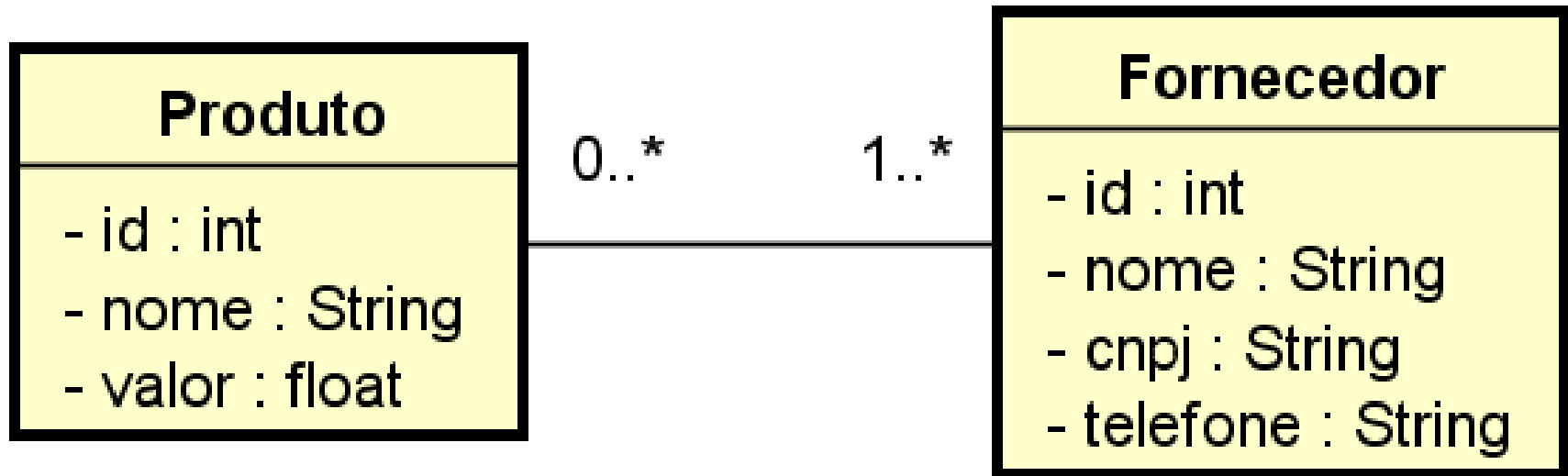
```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private List<Modelo> modelos;  
    ...  
}
```

Muitos para muitos (many-to-many)

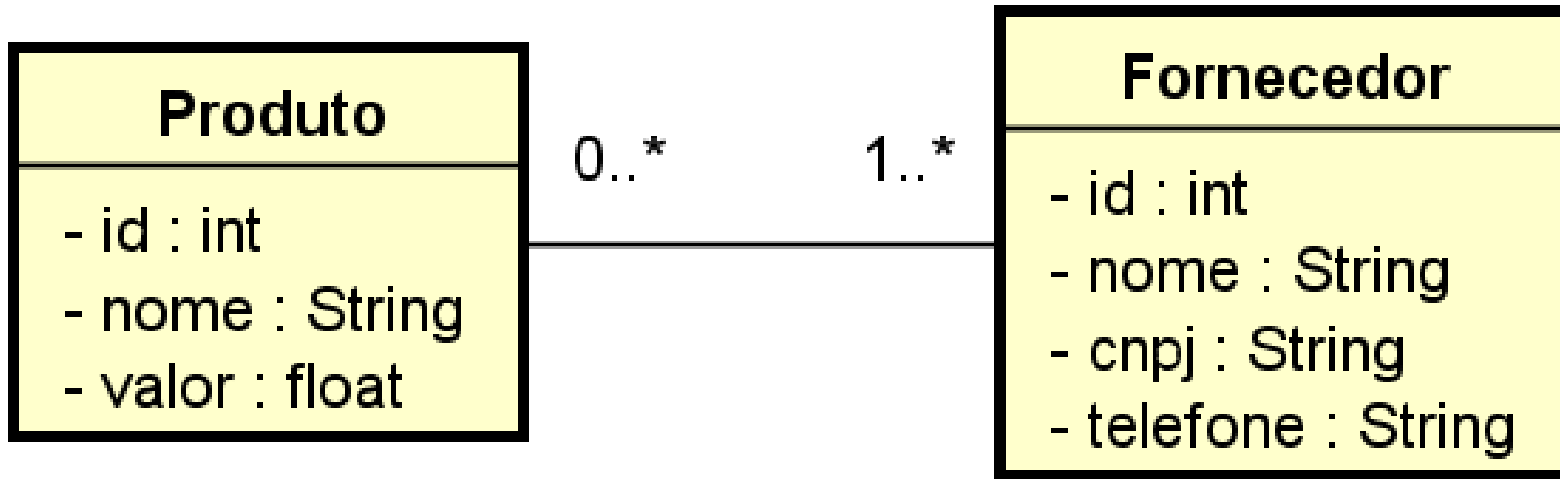
- Cada objeto de uma classe pode estar associado a muitos de outra classe, e vice-versa
- Exemplos
 - Um mecânico pode consertar vários carros e um carro pode ser consertado por vários mecânicos
 - Um aluno pode estar matriculado em várias disciplinas, e cada disciplina pode possuir vários alunos matriculados



Muitos para muitos (many-to-many)



Muitos para muitos (many-to-many)



```
public class Produto {  
    private int id;  
    private String nome;  
    private float valor;  
    private List<Fornecedor>  
        fornecedores;  
    ...  
}
```

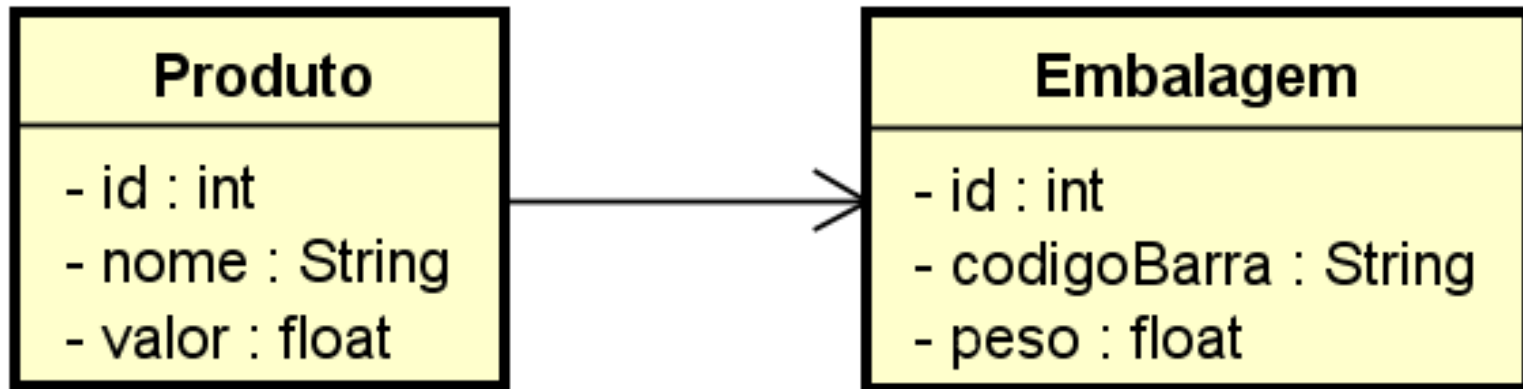
```
public class Fornecedor {  
    private int id;  
    private String nome;  
    private String cnpj;  
    private String telefone;  
    private List<Produto> produtos;  
    ...  
}
```

Associações no JPA

- @OneToOne (associações UM PARA UM)
- @OneToMany (associações UM PARA MUITOS)
- @ManyToOne (associações MUITOS PARA UM)
- @ManyToMany (associações MUITOS PARA MUITOS)



JPA - Associação um para um



JPA - Associação um para um

```
@Entity
public class Produto {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private float valor;

    @OneToOne
    @JoinColumn(unique = true)
    private Embalagem embalagem;
```

```
@Entity
public class Embalagem {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String codigoBarra;
    private float peso;

    ....
}
```

...



INSTITUTO FEDERAL
Rio Grande do Sul



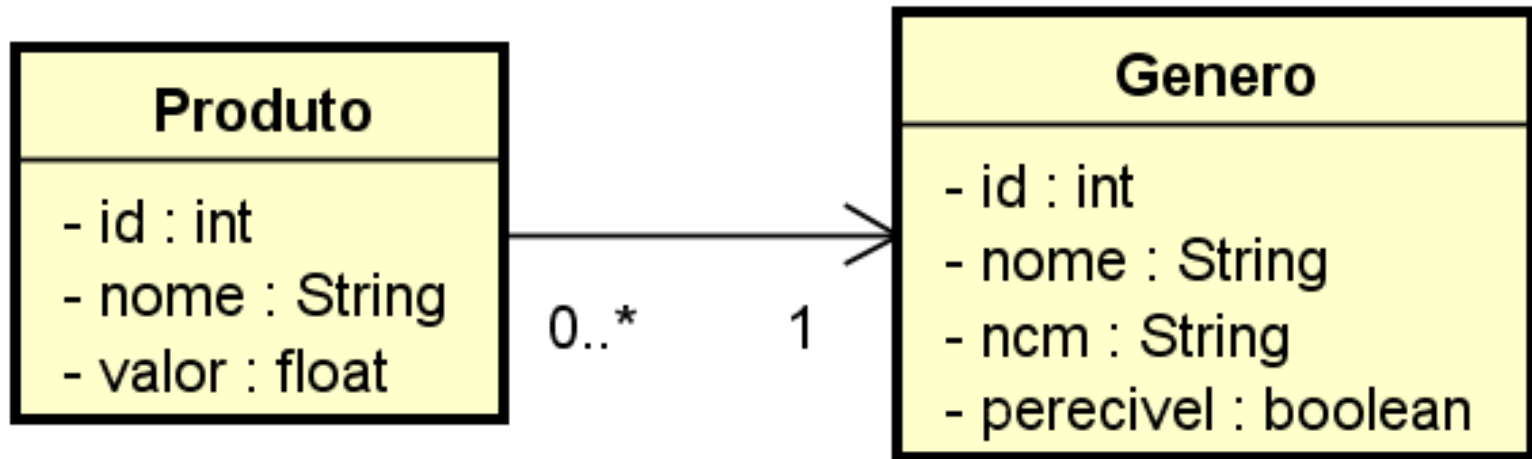
JPA - Associação um para um

v	ds1	produto
id	:	int(11)
nome	:	varchar(255)
valor	:	float
embalagem_id	:	int(11)

v	ds1	embalagem
id	:	int(11)
codigo_barra	:	varchar(255)
peso	:	float



JPA - Associação muitos para um



JPA - Associação muitos para um

```
@Entity
public class Produto {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private float valor;

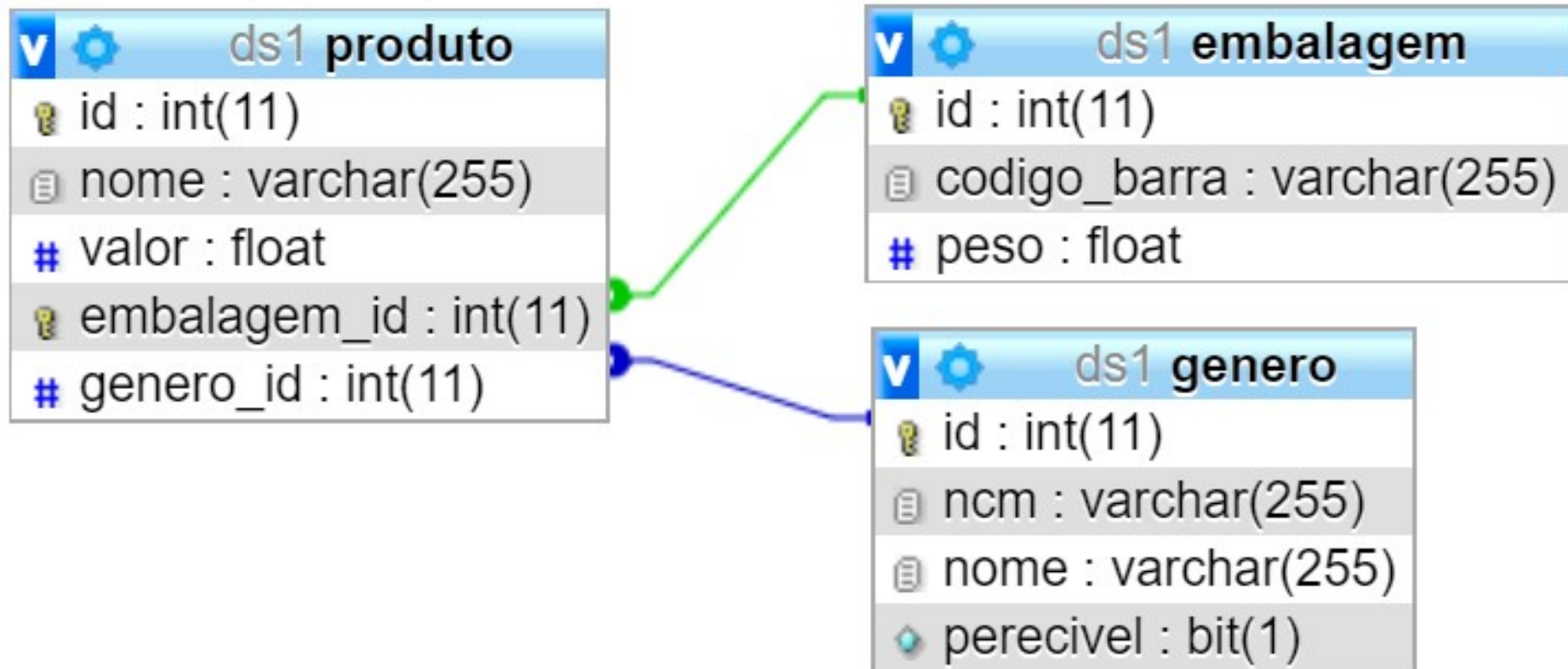
    @ManyToOne
    private Genero genero;
```

```
@Entity
public class Genero {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private String ncm;
    private boolean perecivel;

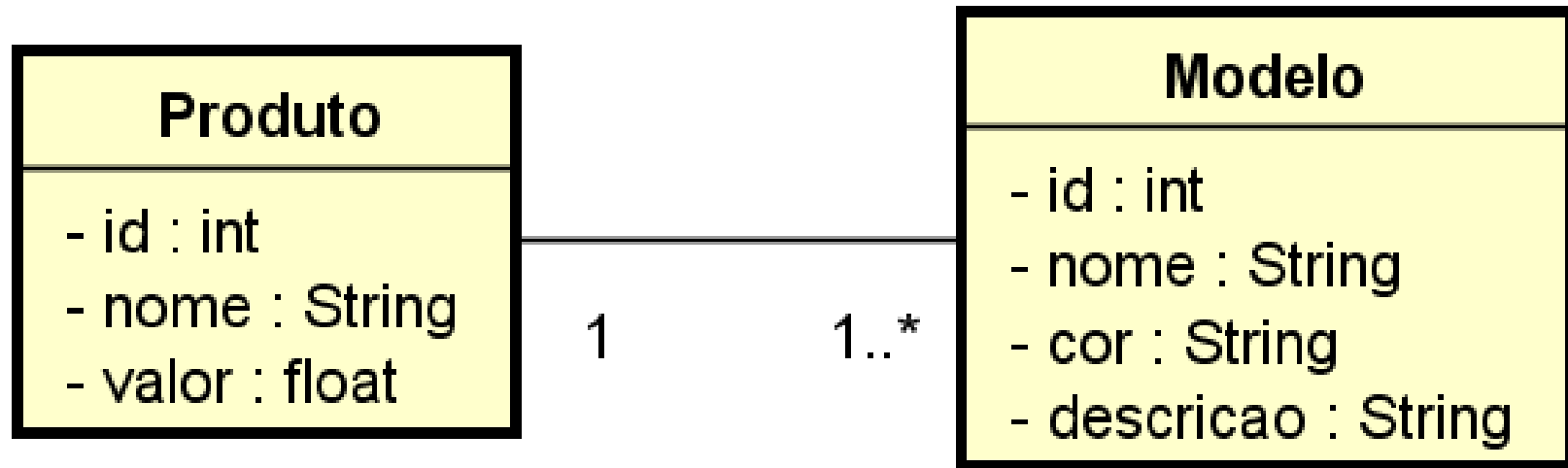
    ....
}
```



JPA - Associação muitos para um



JPA - Associação um para muitos



JPA - Associação um para muitos

```
@Entity
public class Produto {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private float valor;

    @OneToMany
    private List<Modelo> modelos;
```

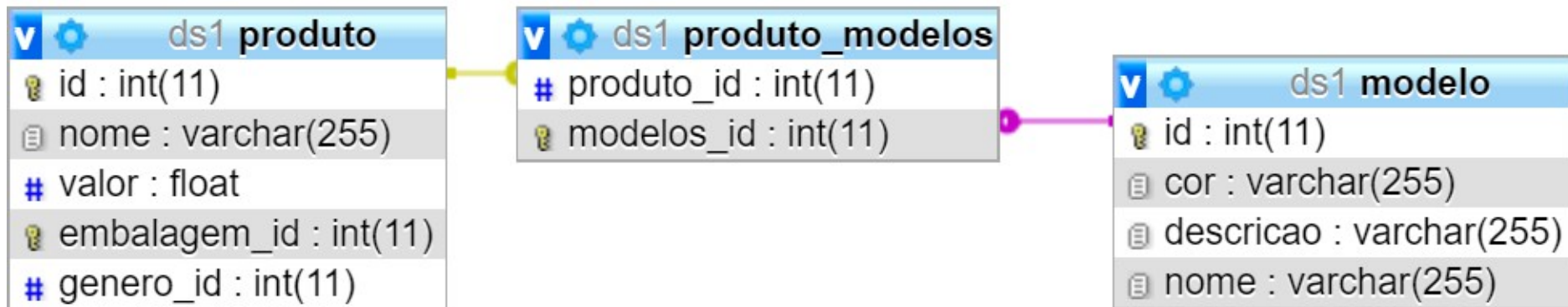
```
...
}
```

```
@Entity
public class Modelo {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private String cor;
    private String descricao;

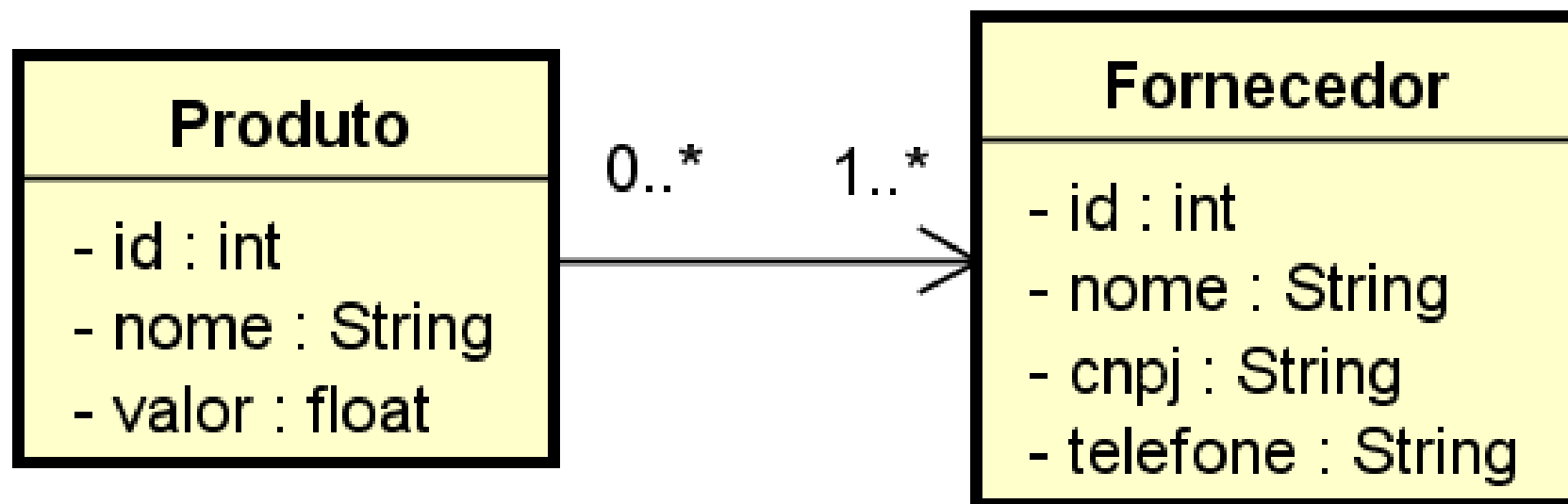
    ...
}
```



JPA - Associação um para muitos



JPA - Associação muitos para muitos



JPA - Associação muitos para muitos

```
@Entity
public class Produto {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private float valor;

    @ManyToMany
    private List<Fornecedor>
        fornecedores;
```

```
@Entity
public class Fornecedor {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private String cnpj;
    private String telefone;

    ...
}
```

```
...
}
```



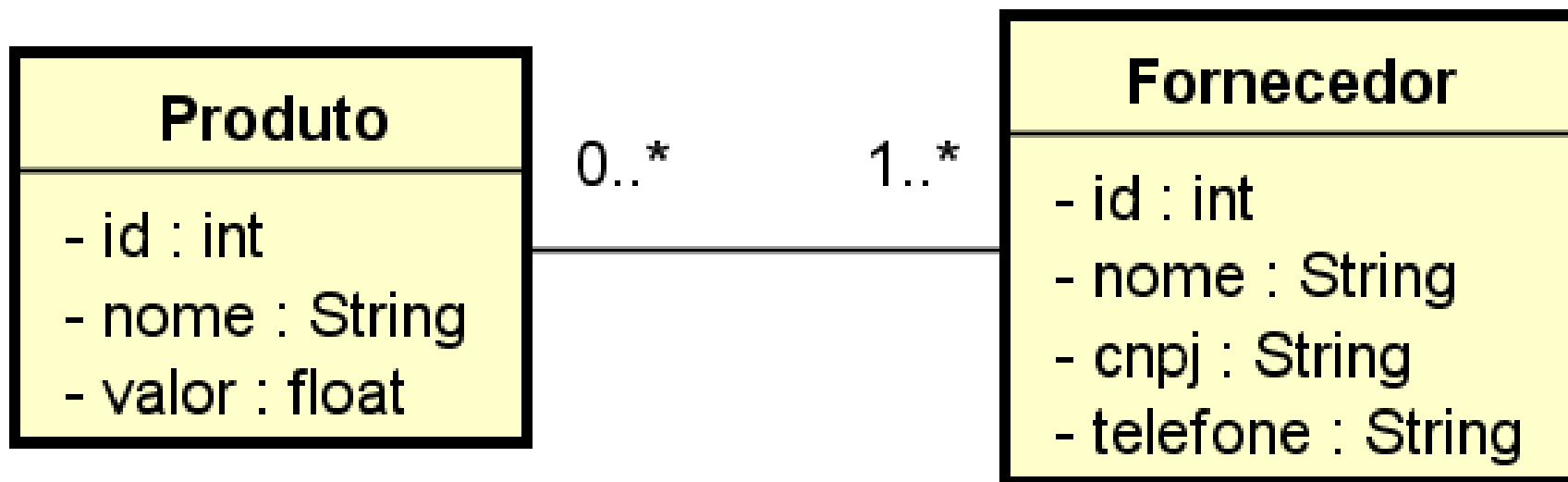
JPA - Associação muitos para muitos

v	ds1	produto
🔑		id : int(11)
📄		nome : varchar(255)
#		valor : float
🔑		embalagem_id : int(11)
#		genero_id : int(11)

v	ds1	produto_fornecedores
#		produto_id : int(11)
#		fornecedores_id : int(11)

v	ds1	fornecedor
🔑		id : int(11)
📄		cnpj : varchar(255)
📄		nome : varchar(255)
📄		telefone : varchar(255)

JPA – bidirecional



JPA – bidirecional

- Para associações bidirecionais no Hibernate se utiliza o atributo *mappedBy*, no elemento não dominante da associação



JPA – bidirecional

```
@Entity
public class Produto {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private float valor;

    @ManyToMany
    private List<Fornecedor>
        fornecedores;
```

```
...
}
```



```
@Entity
public class Fornecedor {
    @Id
    @GeneratedValue(strategy =
        GenerationType.IDENTITY)
    private int id;
    private String nome;
    private String cnpj;
    private String telefone;

    @ManyToMany(
        mappedBy = "fornecedores")
    private List<Produto> produtos;
```

```
...
}
```



Problema de desempenho

- @OneToMany(fetch = FetchType.EAGER)
 - Sempre carrega os dados
- @OneToMany(fetch = FetchType.LAZY)
 - Carrega os dados somente quando acessados



JPA – Associação

Agregação Simples X Composição

Representa o tipo de vínculo entre as partes

Agregação Simples

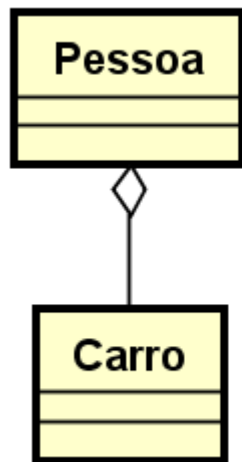
- Vínculo fraco entre as partes, os ciclos de vida dos objetos são independentes

Agregação por Composição

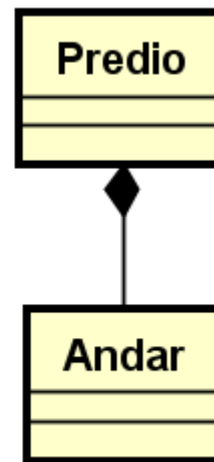
- Vínculo forte entre as partes, o ciclo de um objeto está atrelado a outro



JPA – Associação



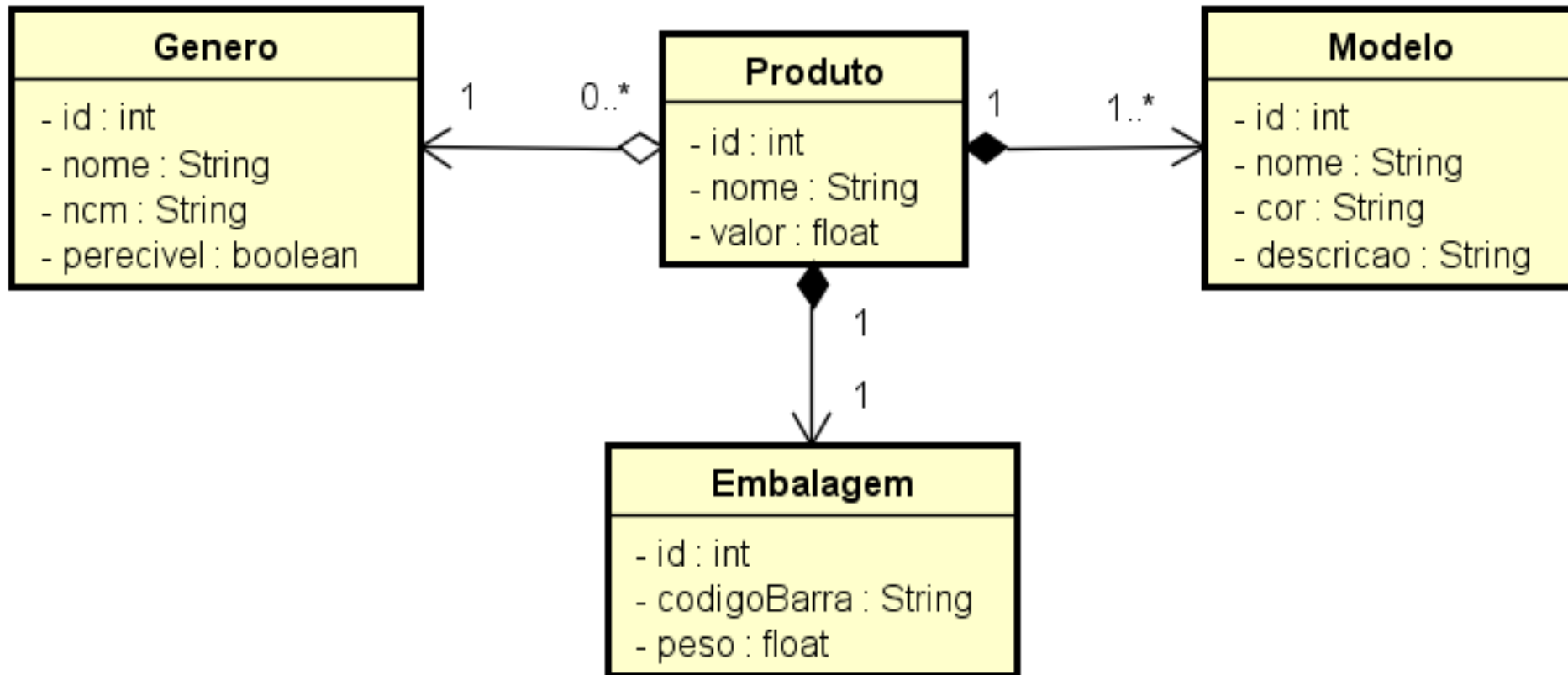
**Agregação
Simples**



**Agregação
por Composição**



JPA – Associação



JPA – Associação

@Entity

```
public class Produto {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private int id;
```

```
    private String nome;
```

```
    private float valor;
```

```
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)
```

```
    @JoinColumn(unique = true)
```

```
    private Embalagem embalagem;
```

```
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
```

```
    private List<Modelo> modelos;
```



Migrando para /api/

- Um servidor REST e um servidor de páginas estático HTML/JavaScript podem ficar no mesmo servidor
- Com o crescimento do número de API REST pode ocorrer conflitos entre os recursos REST e os arquivos html
- /imagens/ é uma pasta contendo imagens ou um recurso REST que gerencia imagens



Migrando para /api/

- Para evitar esse problemas uma boa pratica é deixar todos os recursos REST em uma sub-url
- /api/produtos/
- /api/generos/
- /api/fornecedores/



Migrando para /api/

```
@RestController
@RequestMapping(path = "/api")
public class Produtos {

    @RequestMapping(path= "/produtos/", method =RequestMethod.GET)
    @ResponseStatus(HttpStatus.OK)
    public Iterable<Produto> listar() {
        return produtoDAO.findAll();
    }

    @RequestMapping(path ="/produtos/{id}", method =RequestMethod.C)
    @ResponseStatus(HttpStatus.OK)
    public Produto recuperar(@PathVariable int id) {
```



Associação com SPRING e REST

- O *@RestController* trabalha de forma transparente com associações, mas possui algumas restrições



Associação com SPRING e REST

- Não funciona com associações bidirecionais, caso seja necessário utilizar a associação não dominante deve usar a anotação *@JsonIgnore*



Associação com SPRING e REST

@Entity

```
public class Fornecedor {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private int id;
```

```
    private String nome;
```

```
    private String cnpj;
```

```
    private String telefone;
```

```
    @JsonIgnore
```

```
    @ManyToMany(mappedBy = "fornecedores")
```

```
    private List<Produto> produtos;
```

```
    ...
```

```
}
```



Associação com SPRING e REST

- É possível emular a associação bidirecional usando uma pesquisa

@Repository

public interface ProdutoDAO

extends CrudRepository<Produto, Integer> {

Iterable<Produto> findByGenero(Genero genero);

}



Associação com SPRING e REST

- Para economizar recurso de rede, deve-se evitar a exibição de associações em lista na api GET /item
- Utilizando a api GET /item/{id}/itemLista
- Para isso pode ser usada a notação json:
@JsonProperty(access =
JsonProperty.Access.WRITE_ONLY)
- Que define o campo como somente escrita
- Ou o próprio *@JsonIgnore*



Associação com SPRING e REST

```
public class Produto {  
    ...  
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)  
    @JoinColumn(unique = true)  
    private Embalagem embalagem;  
    @ManyToOne  
    private Genero genero;  
  
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)  
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)  
    private List<Modelo> modelos;  
  
    @JsonIgnore  
    @ManyToMany  
    private List<Fornecedor> fornecedores;
```



Associação com SPRING e REST

- [GET] /api/produtos/1

```
{  
  "id": 1,  
  "nome": "Tomate",  
  "valor": 10.4,  
  "embalagem": {"id": 1, "codigoBarra": "65454", "peso": 10},  
  "genero": {  
    "id": 1,  
    "nome": "Fruta",  
    "ncm": "2434",  
    "perecivel": true  
  }  
}
```



Associação com SPRING e REST

- [GET] /api/produtos/1/modelos/

```
[  
  {  
    "id": 1,  
    "nome": "Italiano",  
    "cor": "Vermelha",  
    "descricao": "Tem um formato alongado"  
  },  
  {  
    "id": 2,  
    "nome": "Gaúcho",  
    "cor": "Vermelha",  
    "descricao": "Costuma ser graúdo"  
  }  
]
```

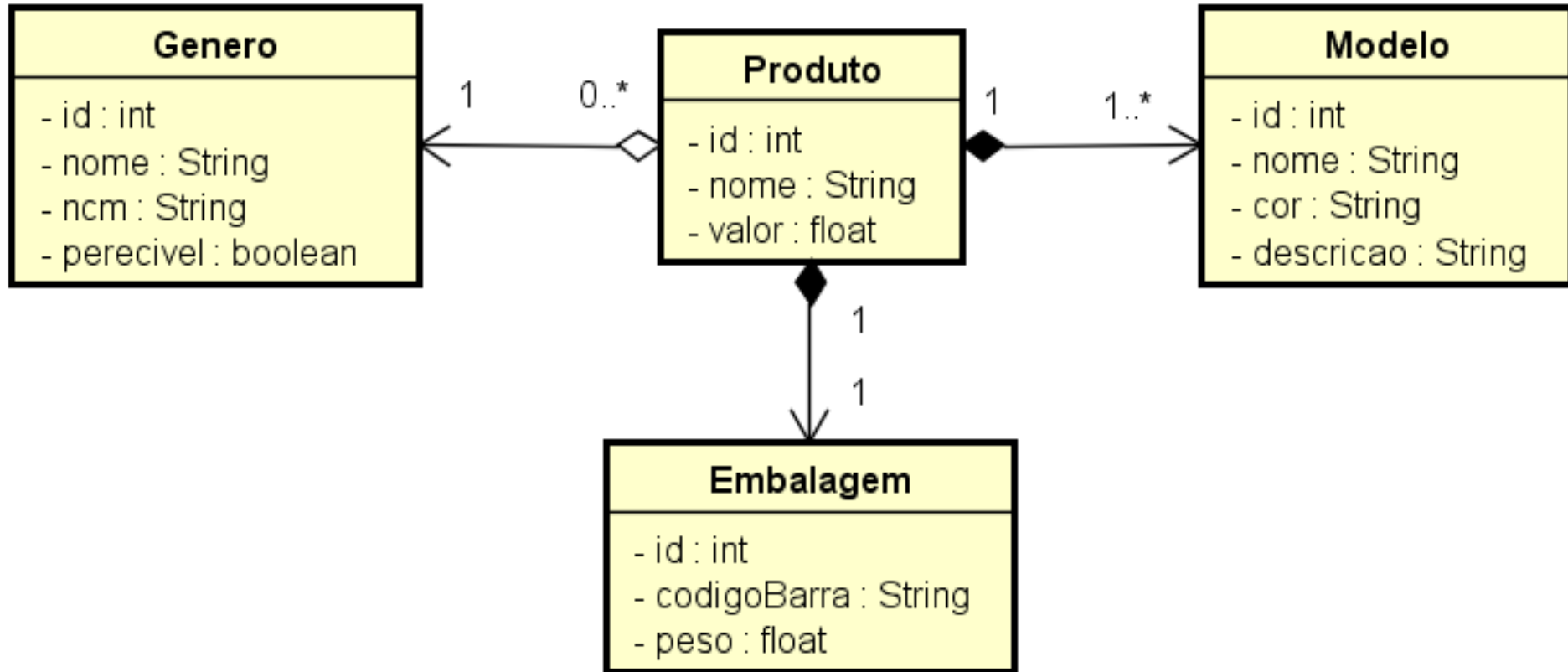


Associação com SPRING e REST

- Em multiplicidades de um para um as entidades podem ser cadastradas juntas
- Em associação por agregação simples cada entidade da associação deve ser cadastrada em seu próprio recurso REST
- Em agregação por composição a entidade pode ser cadastrada no mesmo recurso REST



Associação com SPRING e REST

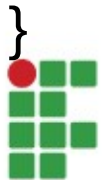


Associação com SPRING e REST

Um para um

- [POST] /api/produtos/
[corpo]

```
{  
  "nome": "Tomate",  
  "valor": 10.40,  
  "embalagem":  
    {  
      "codigoBarra": "65",  
      "peso": 15.0  
    }  
}
```



Associação com SPRING e REST

[resposta]

```
{
  "id": 1,
  "nome": "Tomate",
  "valor": 10.4,
  "embalagem": {
    "id": 1,
    "codigoBarra": "65",
    "peso": 15
  },
  "genero": null
}
```



Associação com SPRING e REST

Agregação simples

- [POST] /api/generos/

[corpo]

```
{  
  "nome": "Fruta",  
  "ncm": "2434",  
  "perecivel": true  
}
```



Associação com SPRING e REST

Agregação simples

- [POST] /api/generos/
[resposta]

```
{  
  "id": 1,  
  "nome": "Fruta",  
  "ncm": "2434",  
  "perecivel": true  
}
```



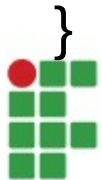
Associação com SPRING e REST

Agregação simples

- [POST] /api/produtos/

[corpo]

```
{  
  "nome": "Abacate",  
  "valor": 9.3,  
  "embalagem": {"codigoBarra": "6324325", "peso": 2},  
  "genero": {  
    "id": 1, "nome": "Fruta",  
    "ncm": "2434", "perecivel": true  
  }  
}
```



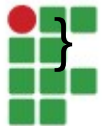
Associação com SPRING e REST

Agregação simples

- [POST] /api/produtos/

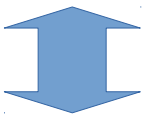
[corpo]

```
{  
  "id": 5,  
  "nome": "Abacate",  
  "valor": 9.3,  
  "embalagem": { "id": 5, "codigoBarra": "6324325", "peso": 2 },  
  "genero": {  
    "id": 1, "nome": "Fruta",  
    "ncm": "2434", "perecivel": true  
  }  
}
```



Associação com SPRING e REST

```
{
  "nome": "Abacate",
  "valor": 9.3,
  "embalagem": {"codigoBarra": "6324325", "peso": 2},
  "genero": {
    "id": 1, "nome": "Fruta",
    "ncm": "2434", "perecivel": true
  }
}
```



```
{
  "nome": "Abacate",
  "valor": 9.3,
  "embalagem": {"codigoBarra": "6324325", "peso": 2},
  "genero": {"id": 1} // Só a id já basta no cadastro
}
```



Associação com SPRING e REST

Agregação simples

- [DELETE] /api/produtos/4
 - Vai excluir a embalagem de id 4, mas não vai excluir o gênero de id 1



Associação com SPRING e REST

Agregação por composição

- [POST] /api/produtos/1/modelos/

[corpo]

```
{  
  "nome": "Gaúcho",  
  "cor": "Vermelha",  
  "descricao": "Costuma ser graúdo"  
}
```



Associação com SPRING e REST

Agregação por composição

- [POST] /api/produtos/1/modelos/
[resposta]

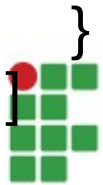
```
{  
  "id": 2,  
  "nome": "Gaúcho",  
  "cor": "Vermelha",  
  "descricao": "Costuma ser graúdo"  
}
```



Associação com SPRING e REST

Agregação por composição

- [GET] /api/produtos/1/modelos/
[
 { "id": 1,
 "nome": "Italiano",
 "cor": "Vermelha",
 "descricao": "Tem um formato alongado"
 },
 { "id": 2,
 "nome": "Gaúcho",
 "cor": "Vermelha",
 "descricao": "Costuma ser graúdo"
 }
]



Associação com SPRING e REST

Agregação por composição

- modelo

id	cor	descricao	nome
1	Vermelha	Tem um formato alongado	Italiano
2	Vermelha	Costuma ser graúdo	Gaúcho

- produto_modelos

produto_id	modelos_id
1	1
1	2



Associação com SPRING e REST

Agregação por composição

- [DELETE] /api/produtos/1/modelos/1
- [GET] /api/produtos/1/modelos/

```
[  
  {  
    "id": 2,  
    "nome": "Gaúcho",  
    "cor": "Vermelha",  
    "descricao": "Costuma ser graúdo"  
  }  
]
```



Associação com SPRING e REST

Agregação por composição

- modelo

id	cor	descricao	nome
2	Vermelha	Costuma ser graúdo	Gaúcho

- produto_modelos

produto_id	modelos_id
1	2

