07

Consolidando o seu conhecimento

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Na sua classe `Ator`, adicione a propriedade `$filmes`.

2) No arquivo de mapeamento da entidade `Ator`, adicione o relacionamento da seguinte forma:

```
<many-to-many field="filmes" target-entity="Alura\Doctrine\Entity\Filme">
    <join-table name="ator_filme">
        <join-columns>
            <join-column name="id_ator" referenced-column-name="id_ator" />
        </join-columns>
        <inverse-join-columns>
            <join-column name="id_filme" referenced-column-name="id_filme"/>
        </inverse-join-columns>
    </join-table>
</many-to-many>
```

3) Agora, na classe `Filme`, adicione o atributo `$atores`.

4) No arquivo de mapeamento da entidade `Filme`, faça o relacionamento com:

```
<many-to-many field="atores"
              target-entity="Alura\Doctrine\Entity\Ator"
              mapped-by="filmes" />
```

5) Adicione o seguinte código à sua classe `Ator` (não se esqueça de importar `Doctrine\Common\Collection\ArrayCollection`):

```
public function __construct(
    ?int $id,
    string $primeiroNome,
    string $ultimoNome
) {
    $this->id = $id;
    $this->primeiroNome = $primeiroNome;
    $this->ultimoNome = $ultimoNome;
    $this->ultimaAtualizacao = new \DateTimeImmutable();
    $this->filmes = new ArrayCollection();
}

public function addFilme(Filme $filme): void
{
    if ($this->filmes->contains($filme)) {
        return;
    }

    $this->filmes->add($filme);
    $filme->addAtor($this);
}
```

6) Agora, na sua classe `Filme`, adicione o seguinte código (não se esqueça de importar `Doctrine\Common\Collection\ArrayCollection`):

```
public function __construct(
    ?int $id,
    string $titulo,
    ?string $sinopse = null,
    ?string $anoLancamento = null
) {
    $this->id = $id;
    $this->titulo = $titulo;
    $this->sinopse = $sinopse;
    $this->anoLancamento = $anoLancamento;
    $this->ultimaAtualizacao = new \DateTimeImmutable();
    $this->atores = new ArrayCollection();
}

public function addAtor(Ator $ator): void
{
    if ($this->atores->contains($ator)) {
        return;
    }

    $this->atores->add($ator);
    $ator->addFilme($this);
}
```

7) Na raiz do projeto, crie um arquivo chamado `teste.php`, com o seguinte conteúdo:

```
use Alura\Doctrine\Entity\Ator;
use Alura\Doctrine\Entity\Filme;
use Alura\Doctrine\Entity\Idioma;

require_once 'vendor/autoload.php';

$em = (new \Alura\Doctrine\Helper\EntityManagerCreator())->criaEntityManager();
$ator = new Ator(null, 'Vinicius', 'Dias');

$filme1 = new Filme(null, 'A volta dos que não foram');
$filme2 = new Filme(null, 'As longas tranças do careca');

$ator->addFilme($filme1);
$ator->addFilme($filme2);

$em->persist($filme1);
$em->persist($filme2);
$em->persist($ator);
$em->flush();
```

8) Utilizando o `pgAdmin`, crie um novo banco de dados, chamado `alura_filmes_novo`. Aproveite para editar o nome do banco em seu `EntityManagerCreator`.

9) Na linha de comando, execute `vendor\bin\doctrine orm:schema-tool:create`. Com o banco de dados criado, execute `php teste.php`. Garanta que o novo ator e os filmes foram salvos.