

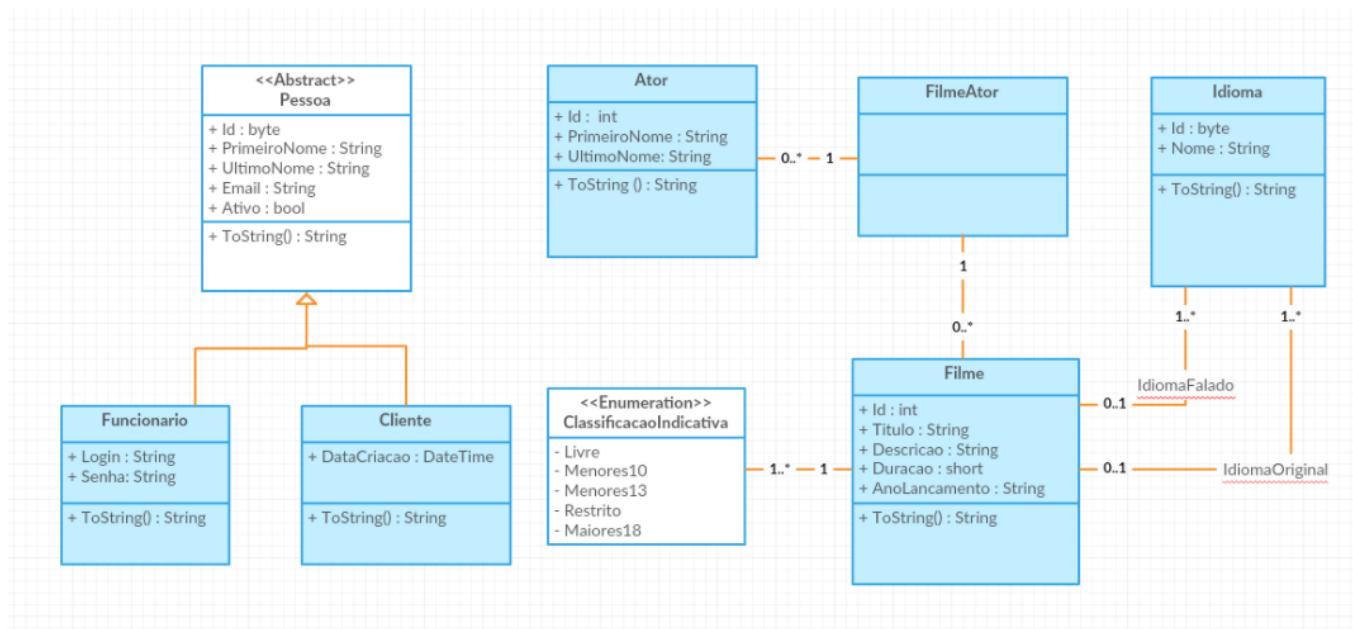
04

## Mapeando atores

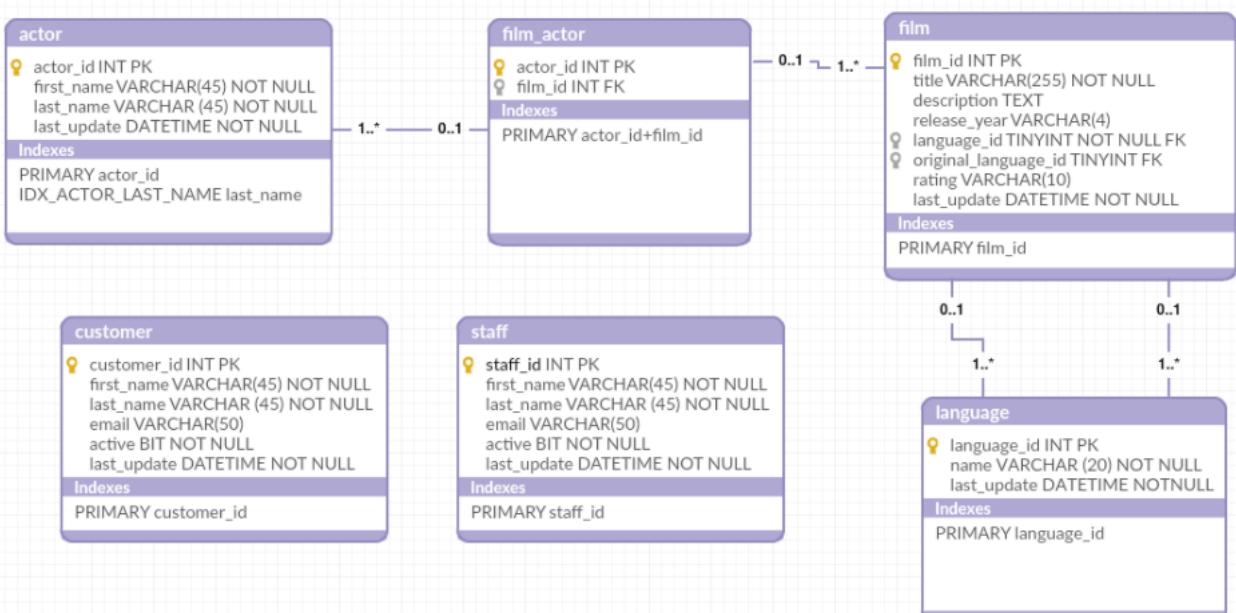
### Transcrição

A nossa primeira tarefa nesta aula é mapear as classes do diagrama da nossa aplicação "Alura Filmes", para as tabelas do banco de dados.

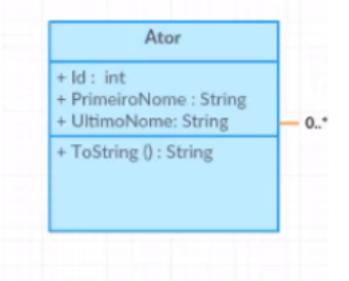
Abaixo temos nosso diagrama de classes:



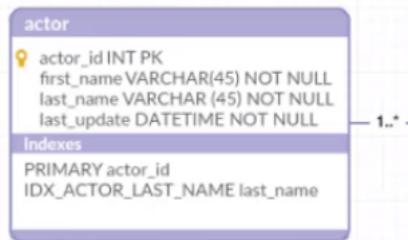
As entidades a serem relacionadas:



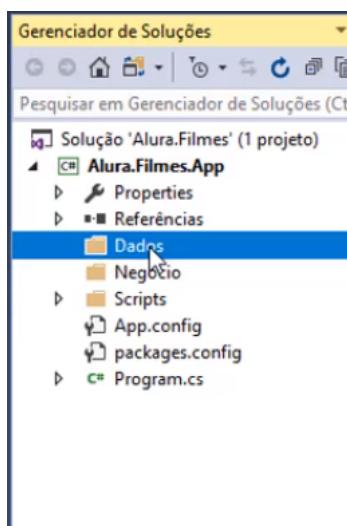
Primeiramente selecionaremos a classe **Autor** ...



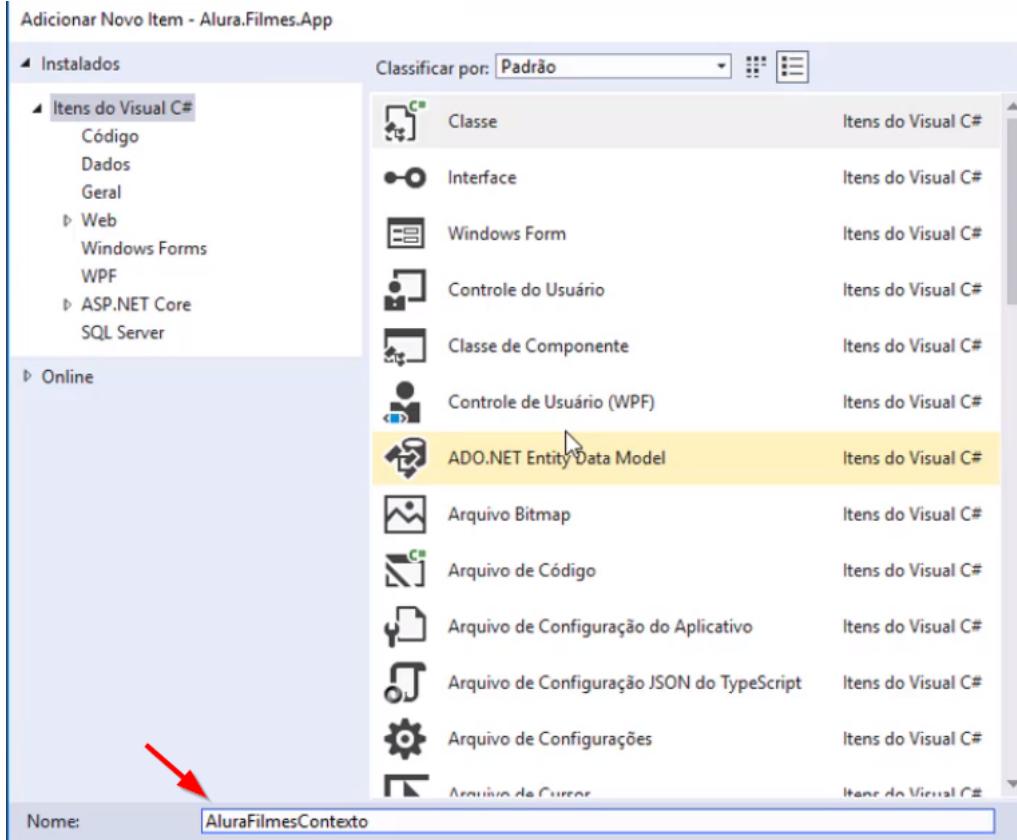
Para em seguida, mapeá-la à tabela `actor`.



Abriremos o Visual Studio para executar as próximas etapas. Como queremos integrar as informações ao Entity, precisamos criar uma classe de contexto. Colocaremos a nossa classe dentro da pasta "Dados". Na área do "Gerenciador de Soluções", clicaremos com o botão direito na pasta e selecionaremos "Adicionar > Classe".



A classe se chamará `AluraFilmesContexto`.



A classe contexto precisa relacionar três informações: a primeira é o Entity. Isso será feito por meio da herança com a classe `DbContext`.

```
namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
    }
}
```

A segunda informação é a conexão com o banco de dados, ou seja, qual banco de dados o Entity irá se direcionar. Uma das maneiras de realizar esse direcionamento é utilizando o método `OnConfiguring()`. Adicionaremos, ainda, o `optionsBuilder` para indicar que usaremos o `SqlServer`, que foi o *provider* instalado. Utilizamos a string de conexão que usamos como argumento do método `SqlServer`.

```
namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=AluraFilmes;Trusted_C
        }
    }
}
```

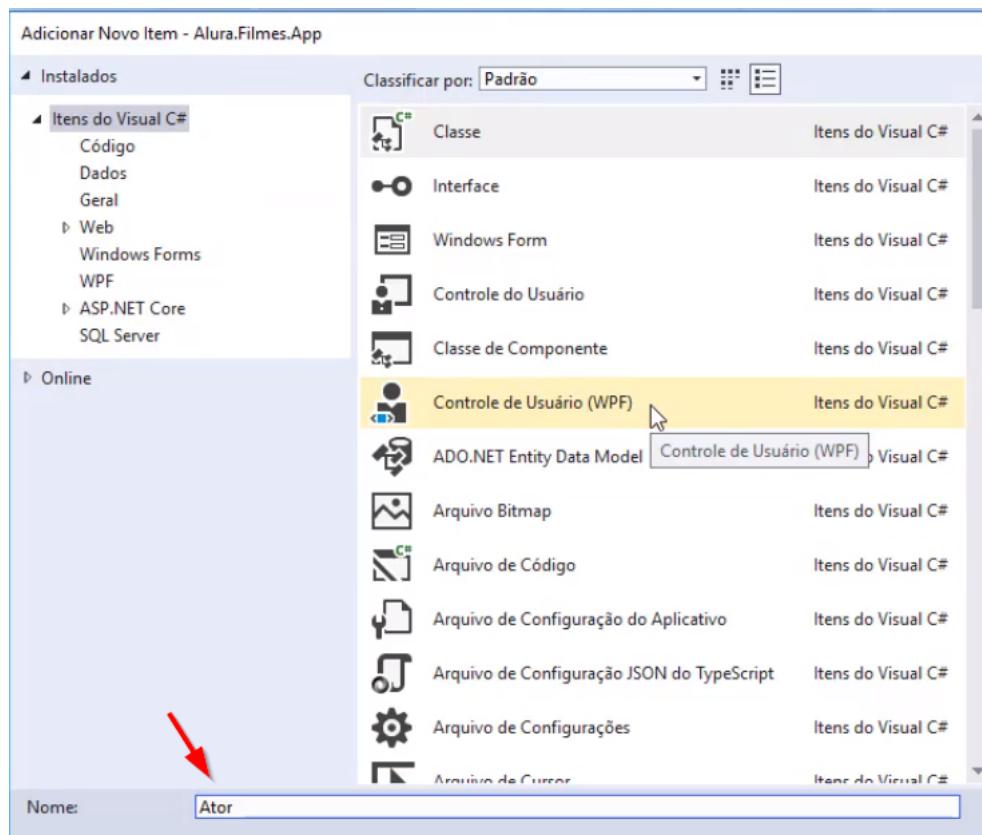
A terceira informação é especificar quais propriedades devem ser gerenciadas pelo Entity. Faremos isso usando a propriedade `DbSet` para classe `Autor`. Chamaremos a propriedade de `Atores`.

```
using Microsoft.EntityFrameworkCore;

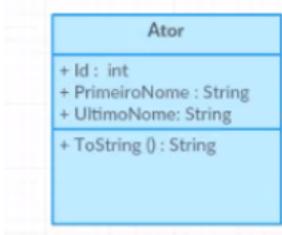
namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
        public DbSet<Autor> ATORES { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=AluraFilmes;Trusted_Connection=True");
        }
    }
}
```

A classe `Autor` ainda não existe. Iremos colocar todas as classes de negócio na pasta "Negocio", localizada na área "Gerenciador de Soluções". Clicaremos sob a pasta com o botão direito e selecionaremos "Adicionar > Classe". Criamos, então, a classe `Autor`.



Ao criar a classe `Autor`, devemos nos lembrar das propriedades do diagrama de classes: `Id`, `PrimeiroNome` e `UltimoNome`. Faremos elas, portanto.

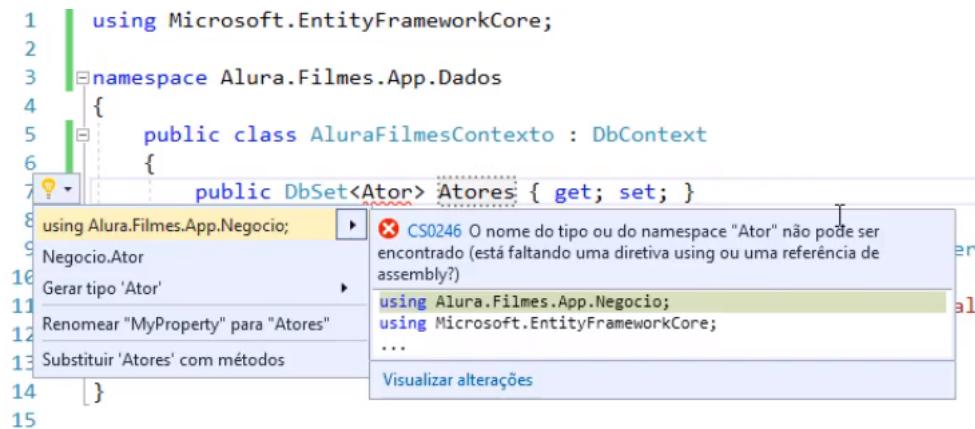


```

namespace Alura.Filmes.App.Negocio
{
    public class Ator
    {
        public int Id {get; set;}
        public string PrimeiroNome { get; set; }
        public string UltimoNome { get; set; }
    }
}

```

Importaremos a classe para o contexto. Com isso, ele irá conter as três informações necessárias.



Neste ponto, podemos realizar um teste básico para sabermos se os registros da tabela de atores pode ser acessado. O teste será um `select em Program`. Para realizar isso usando o Entity, criaremos uma instância daquele contexto, no caso, será `AluraFilmesContexto`.

```

namespace Alura.Filmes.App
{
    class Program
    {
        static void Main(string[] args)
        {
            //select * from actor
            using (var contexto = new AluraFilmesContexto())
            {

            }
        }
    }
}

```

Faremos o `foreach` que irá percorrer a propriedade `Atores`. Quando a propriedade for percorrida, o Entity irá popula-la e fará isso utilizando `select`. Depois das modificações, iremos iniciar a aplicação.

```
namespace Alura.Filmes.App
{
    class Program
    {
        static void Main(string[] args)
        {
            //select * from actor
            using (var contexto = new AluraFilmesContexto())
            {
                foreach (var ator in contexto.Atores)
                {
                    System.Console.WriteLine();
                }
            }
        }
    }
}
```

Perceberemos que ocorrerá um erro `*invalid object name "Atores"`.

```
C:\Windows\system32\cmd.exe
Exceção Sem Tratamento: System.Data.SqlClient.SqlException: Invalid object name 'Atores'.
   em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   em System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
   em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
   em System.Data.SqlClient.SqlDataReader.get_MetaData()
   em System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption)
   em System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader ds, Boolean describeParameterEncryptionRequest)
   em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Task& task, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry)
   em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
   em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method)
   em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior)
   em System.Data.Common.DbCommand.ExecuteReader()
   em Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand.ExecuteNonQuery(IRelationalConnection connection, DbCommandMethod executeMethod, IReadOnlyDictionary`2 parameterValues)
```

`Atores` é o nome da propriedade que definimos para o `DbSet` da classe `Autor`. Mudaremos o nome da propriedade para `ConjuntoDeAtores` e executaremos novamente o programa.

```
using Alura.Filmes.App.Negocio;
using Microsoft.EntityFrameworkCore;

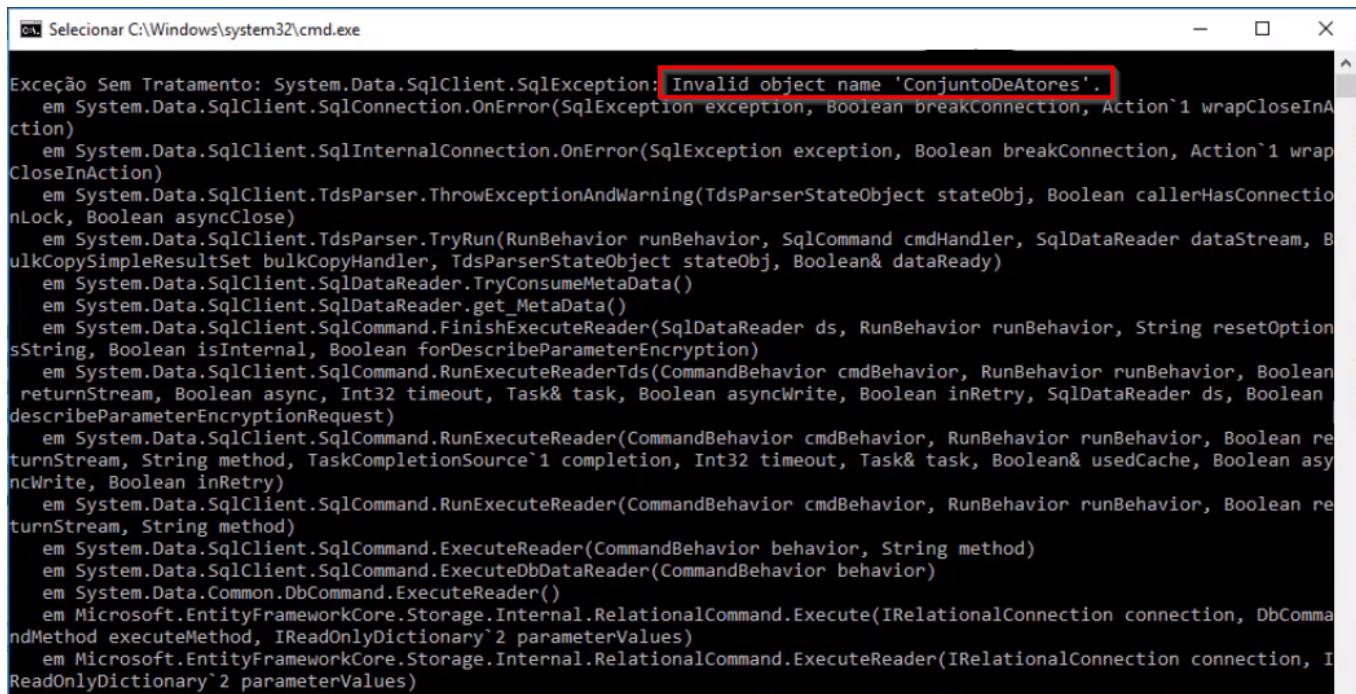
namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
        public DbSet<Aktor> ConjuntoDeAtores { get; set; }
```

```

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer("Server=(localdb)\mssqllocaldb;Database=AluraFilmes;Trusted_Connection=True");
}
}

```

Novamente o programa anunciará um erro. Perceberemos que nesse caso o erro foi `Invalid object name 'ConjuntoDeAtores'`. O Entity, para poder mapear o nome da tabela correspondente à classe `Autor`, está se valendo do nome da propriedade `DbSet` da mesma classe. Essa é uma regra de execução implícita do Entity.



```

Exception Sem Tratamento: System.Data.SqlClient.SqlException: Invalid object name 'ConjuntoDeAtores'.
   em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   em System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
   em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
   em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
   em System.Data.SqlClient.SqlDataReader.get_MetaData()
   em System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption)
   em System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader ds, Boolean describeParameterEncryptionRequest)
   em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Task& task, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry)
   em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
   em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method)
   em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior)
   em System.Data.Common.DbCommand.ExecuteReader()
   em Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand.ExecuteNonQuery(IRelationalConnection connection, DbCommandMethod executeMethod, IReadOnlyDictionary`2 parameterValues)
   em Microsoft.EntityFrameworkCore.Storage.Internal.RelationalCommand.ExecuteReader(IRelationalConnection connection, IReadOnlyDictionary`2 parameterValues)
   ...

```

Renomearemos a propriedade para `Atores`.

```

using Alura.Filmes.App.Negocio;
using Microsoft.EntityFrameworkCore;

namespace Alura.Filmes.App.Dados
{
    public class AluraFilmesContexto : DbContext
    {
        public DbSet<Atores> Atores { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=(localdb)\mssqllocaldb;Database=AluraFilmes;Trusted_Connection=True");
        }
    }
}

```

Denominam-se "convenções" as regras implícitas de execução do Entity. O software utiliza um sistema de convenções para simplificar o trabalho do desenvolvedor, ou seja, ele não precisa mapear todas as classes e propriedades, basta que você siga