Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2020/2021

Séries & Companhia

Ana Carneiro, Ana Peixoto, Luís Pinto, Pedro Fernandes

Dezembro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Séries & Companhia

Ana Carneiro, Ana Peixoto, Luís Pinto, Pedro Fernandes

Dezembro, 2020

Resumo

No âmbito da disciplina de Bases de Dados, foi desenvolvido o projeto de gestão de BD aplicada á plataforma de *streaming*, Séries & Companhia. Esta aplicação, tal como muitas outras no mercado, dedica-se a fornecer séries internacionalmente conhecidas a qualquer um que decida subscrever aos seus serviços. Para além disso, esta ainda fornece informações extra sobre as séries contidas no seu cardápio.

A inicialização do projeto passou por levantar os requisitos necessários para a implementação de uma base de dados coesa e relevante para o caso em estudo. De seguida, avaliaram-se quais a entidades pertinentes para o projeto assim como os seus respetivos atributos e tipo de dados, de forma a construir um modelo conceptual representativo do problema em questão. Posteriormente, construiu se o modelo lógico com a ajuda da plataforma *MySQL Workbench*. Nesta fase também foram implementadas as interrogações necessárias ao cumprimento dos requisitos propostos através de álgebra relacional. Finalmente, implementou-se o modelo físico com ajuda de ferramentas do *MySQL Workbench* (*foward engineering*, para gerar o código da BD) e as interrogações realizadas em linguagem SQL. Para além disso, também foram criados índices e vistas que visam melhorar a gestão e administração da BD implementada e analisou-se o seu armazenamento atual assim como o crescimento anual espectável da mesma.

Após as diversas reuniões no fim de cada etapa que tinham como objetivo validar o trabalho realizado até então, conseguimos resolver todos os problemas de forma estruturada e organizada que culminou na conclusão do projeto.

Área de Aplicação: Desenho, desenvolvimento e implementação de Sistemas de Bases de Dados.

Palavras-Chave: Bases de Dados, Álgebra Relacional, Entidades Atributos, Relações, SQL, *MySQL Workbench*, Modelo Conceptual, Modelo Lógico, Implementação Físico, Séries & Companhia, Índice, Vista.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Historicamente, temos vindo a consumir conteúdo em três plataformas. Assistimos ao cinema desde a década de 1880, à televisão desde a década de 1930 e às plataformas de *streaming* digital desde 2000. As plataformas de *streaming* em Portugal tem crescido nos últimos anos e, entre pequenas e grandes plataformas, existem já cerca de uma dezena de serviços de *streaming* no nosso país, com mais de 1 milhão e meio de subscritores em agosto de 2019. Foi neste verão de 2019 que nasceu a plataforma **Séries & Companhia**, uma plataforma de *streaming* exclusiva para séries.

Com o surgimento da pandemia do COVID-19 e consequente confinamento social, milhões de pessoas ficaram presas em casa, o que levou a uma maior procura por entretenimento "sem sair de casa". Isto fez com que os serviços de *streaming* tivessem um aumento do fluxo de subscritores e do seu número total de espectadores. Foi neste novo ambiente que a plataforma **Séries & Companhia** ganhou novos subscritores que procuram entretenimento nestes tempos difíceis de pandemia.

1.2. Apresentação do Caso de Estudo

A plataforma **Séries & Companhia**, sediada em Braga, foi fundada por alunos da universidade do Minho do 3º ano de Engenharia Informática no verão de 2019. Esta plataforma tem como intuito fazer chegar à casa dos portugueses as melhores séries internacionais, numa plataforma unificada.

Desde que alguém subscreve ao nosso serviço, garantimos a melhor qualidade de vídeo e áudio possíveis até à data, isto tudo num pacote bastante económico, destacando-se da concorrência. Esta plataforma possui uma lista diversificada de séries para todos os gostos permitindo que cada utilizador, siga "a par e passo" as histórias das suas personagens favoritas.

Comprometemo-nos a fazer companhia a todos que usufruírem do nosso serviço, daí o nome **Séries & Companhia.**

1.3. Motivação e Objetivos

Com o início da situação pandémica atual, mais utilizadores começaram a utilizar a aplicação de *streaming* **Séries & Companhia**. Com este aumento da procura, a plataforma deixou de conseguir cumprir os requisitos necessários ao bom funcionamento da mesma, devido á maneira elementar com que estava a lidar com os dados armazenados. Houve necessidade de melhorar a qualidade de serviço da aplicação e que esta tivesse a capacidade de catalogar e organizar cada vez mais séries de forma a que os seus subscritores tivessem uma experiência mais completa.

Devido a estas razões optou-se por criar um Sistema de Gestão de Bases de Dados mais completo e eficiente. A criação desta base de dados vai nos permitir aumentar o número de séries no catálogo, melhorar a sua organização e acessibilidade e conseguir armazenar informação extra sobre as séries existentes. Assim, é possível que o utilizador consiga efetuar operações que anteriormente não eram possíveis como consultar o *top* das séries mais vistas, avaliar as séries que viu e ver a avaliação de outros utilizadores, ter a capacidade de adicionar uma série á lista de favoritos e ter acesso a informação sobre cada série como o elenco principal, os criadores, o número de prémios recebidos, entre outras. Além disso, para os gestores da aplicação, esta nova BD facilita a consulta de informação acerca dos utilizadores e das distribuidoras de séries, tornando o armazenamento mais eficiente e organizado e permitindo catalogar cada série conforme o seu género.

Deste modo, conseguimos dar uma melhor resposta à crescente procura verificada, oferecer uma experiência mais completa aos utilizadores da plataforma e, principalmente, ter um sistema de armazenamento de dados mais funcional, consistente e eficiente, capaz de sustentar a empresa durante os próximos anos.

1.4. Análise de Viabilidade

Com o investimento num novo Sistema de Bases de Dados Relacional, esperamos obter um melhor serviço que proporcionará uma melhor experiência aos utilizadores, que por sua vez se traduzirá num aumento da procura por parte dos consumidores e, consequentemente, numa maior receita para a empresa. Dado que não foi necessário adquirir um serviço de uma empresa externa para a criação da BD, espera-se que os gastos sejam reduzidos. Concluímos assim que a receita gerada irá cobrir os custos de implementação, tratando-se de investimento favorável com uma boa margem de lucro.

1.5. Estrutura do Relatório

O presente relatório está estruturado em 7 secções: Definição do Sistema, Levantamento e Análise de Requisitos, Modelação Conceptual, Modelação Lógica, Implementação física, Conclusões e trabalho futuro e Referências Bibliográficas.

A **Definição do Sistema** introduz uma pequena apresentação das funcionalidades da plataforma Séries & Companhia, as razões da criação de uma BD a ser implementada na aplicação e também o contexto em qual aplicação esta inserida.

No caso do **Levantamento e Análise de Requisitos**, foi elaborada uma lista com os requisitos das diversas categorias e de seguida a discussão dos eventuais problemas encontrados na análise.

Em relação à **Modelação Conceptual**, foram analisadas as entidades, atributos e os seus relacionamentos que culminou na criação do modelo conceptual. Neste capítulo também foram definidos os tipos de dados de cada atributo e o tipo de atributo e definido para cada entidade a sua chave primária.

No capítulo de **Modelação Lógica**, construi-se o modelo lógico, seguindo as regras de derivação e criaram-se as interrogações propostas através de árvores de álgebra relacional.

Na secção da **Implementação Física**, foram implementadas as interrogações, mencionadas anteriormente, em linguagem SQL. Foram ainda mencionados índices e criaram-se vistas. Para além disso, neste capítulo também se estimou a taxa de crescimento da BD anualmente.

Por fim, elaboramos uma apreciação crítica sobre o trabalho realizado, apontando os seus pontos fortes e fracos, na secção **Conclusões e trabalho futuro.**

2. Levantamento e Análise de Requisitos

2.1. Método de levantamento e de análise de requisitos adotados

O método de levantamento de requisitos permite observar e verificar quais funcionalidades que podem ser adicionadas ao sistema. Assim, o levantamento baseou-se em:

- **Observar a concorrência**: através da análise de outras plataformas de *streaming*, foi possível verificar as suas funcionalidades mais populares de modo a perceber quais estavam em falta e valem a pena acrescentar na aplicação.
- **Questionários aos utilizadores**: pelos questionários fornecidos *in App* e *email* aos nossos subscritores, obtemos informações acerca das melhorias e novas funcionalidades que deveriam ser suportadas pelo sistema.
- **Observação direta do sistema**: após análise do nosso próprio sistema, conseguimos verificar potenciais melhorias a implementar.

Deste modo, conseguimos elaborar diversos requisitos relevantes para o nosso sistema, de modo a enriquecê-lo e torná-lo mais funcional e acessível.

2.2. Requisitos levantados

2.2.1. Requisitos de descrição

1. O utilizador pode se registar fornecendo o nome, o cartão para pagamento, a sua morada (localidade, rua, código postal e pais) e o email.
2. O utilizador pode subscrever um serviço e irão ficar registadas as datas de início e fim de subscrição assim como o seu preço.
3. O serviço tem um preço associado, que pode variar ao longo do tempo.
4. O utilizador pode assistir várias séries.
5. Cada série contém várias temporadas.
6. Cada série tem um ou mais criadores.

7. Uma série tem um nome, avaliação, idioma, classificação etária, vários géneros e pode ter n prémios.
8. O serviço consegue transmitir várias séries.
9. Uma série possui um elenco principal, constituído por vários atores.
10. Cada ator é constituído pelo seu nome, nacionalidade, género e data de nascimento.
11. Uma série é fornecida por uma companhia.
12. Uma companhia contém uma data de fundação, nome e comissão (preço a pagar por série da companhia por parte da **Séries & Companhia**).
13. Um episódio é constituído por um nome, avaliação, número, duração e pode ter vários idiomas de legendas.
14. Uma temporada tem um número de temporada e um ano de lançamento.
15. Cada temporada tem vários episódios.
16. Cada utilizador é capaz de marcar uma série como favorita.
17. Um criador é constituído por nome, género, nacionalidade e data de nascimento.

2.2.2. Requisitos de exploração

1. O sistema deverá ser capaz de Identificar qual o top N das séries mais vistas.
2. O sistema deverá ser capaz de identificar qual o top N de séries com melhor classificação.
3. O sistema deverá ser capaz de identificar quais as companhias que forneceram mais séries.
4. O sistema deverá ser capaz de apresentar todas as series e o seu número de prémios.
5. O sistema deverá ser capaz de identificar qual é top N dos utilizadores que assistem mais séries.
6. O sistema deverá ser capaz de identificar as séries de um determinado idioma.
7. O sistema deverá ser capaz de identificar séries cujos episódios possuem legendas de um determinado idioma.
8. O sistema deverá ser capaz de identificar quais as séries apropriadas para uma determinada faixa etária.
9. O sistema deverá ser capaz de identificar quais as séries favoritas de um utilizador.
10. O sistema deverá ser capaz de identificar qual o número de utilizadores com conta ativa.
11. O sistema deverá ser capaz de identificar qual a faturação total.

2.2.3. Requisitos de controlo

1. A plataforma de *streaming* deve estar operacional 24h por dia.
2. Na última terça-feira de cada mês às 5h da manhã deverá ser feita uma cópia de segurança da base de dados.

3. Os pagamentos da subscrição efetuados fora do período das 9h – 19h, em fins de semana ou feriados só serão transacionados no dia útil seguinte.

2.3. Análise e validação geral dos requisitos

Posteriormente ao levantamento dos requisitos, é necessário discuti-los de modo a validá-los. Assim, a discussão dos requisitos foi feita entre os elementos da equipa para resolver possíveis discórdias e ambiguidades em relação à implementação.

Uma das partes mais discutidas foi o atributo **"prémio"** da entidade **"Série".** Esse atributo tem como objetivo representar os prémios de uma determinada série. A dúvida residiu em optar por classificar o atributo como simples e o seu tipo de dados inteiro, ou classificá-lo como multivalorado e opcional. Dado que apenas era necessário consultar o seu número, consideramos que a primeira opção representaria melhor o pretendido, e a questão de opção estaria representada com o seu valor a zero.

Deste modo, foi possível chegar a um consenso entre os elementos da equipa, para evitar ambiguidades ou dúvidas de implementação futuras.

3. Modelação conceptual

3.1. Apresentação da abordagem de modelação realizada

O modelo conceptual surgiu numa primeira instância de modelação. Desta forma, é possível representar as primeiras ilações sobre o sistema a implementar.

Com o auxílio da ferramenta **BRModelo** conseguimos representar esquematicamente um esboço da base de dados, enumerando tudo o que é necessário para a mesma. O esquema conceptual foi obtido através de um diagrama ER (entidade-relação), onde é possível visualizar quais as entidades do sistema, os seus atributos e as relações entre as diferentes entidades. Neste diagrama é possível representar entidades e os seus atributos, discriminando-os por categorias: simples, composto, derivado ou multivalor. Cada entidade representa um objeto do mundo real e as suas caraterísticas. O conjunto de entidades é capaz de sustentar o modelo.

Nesta fase foi possível representar o que foi estipulado na análise de requisitos de uma forma simples e direta. Dada a natureza do esquema conceptual, também foi possível inserir novas ideias de forma prática, facilitando o processo constante de reformulação inerente a uma primeira fase de construção da base de dados.

3.2. Identificação e caracterização das entidades

Entidade	Descrição	Alias	Ocorrência
Série	Entidade que contém informações de uma série.	Programa	A série é o motivo pelo qual os clientes subscrevem o serviço.
Temporada	Entidade que agrega os episódios de uma série.	-	A temporada é a organização da série por episódios.
Ator	Entidade que participa em séries.	Estrela, Personagem, Elenco	O ator é quem faz parte do elenco de uma série.
Companhia	Entidade que fornece séries.	Empresa, Fornecedora, Distribuidora	A companhia é uma empresa que fornece as suas séries em troca de uma comissão.
Criador	Entidade que cria séries.	Realizador	O criador é responsável pela criação da série.
Serviço	Entidade que fornece acesso às séries.	Subscrição	O serviço é uma subscrição que permite ao utilizador assistir séries.
Episódio	Entidade que representa um episódio que o utilizador pode assistir.	-	O episódio é visto pelo utilizador. A duração do episódio pode variar.
Utilizador	Entidade que usufrui do serviço.	Cliente	O utilizador é quem paga pelo serviço e assiste as séries.

Tabela 1 - Tabela de entidades

- **Série**

A entidade **"Série"** é caraterizada por 7 atributos. O "**IDSérie"** trata-se da chave primária desta entidade, capaz de a identificar e distinguir de todas as outras. Esta também possui um **"nome"** que a identifica e refere ao nome da série em questão. O atributo **"avaliação"** reflete a sua classificação e o **"idioma"** a língua original em que foi gravada. A **"classificacaoEtaria"** estabelece um patamar de idade a partir do qual se pode assistir a uma determinada série. No caso do **"género"**, este denota o género cinematográfico em que está inserida, podendo haver mais do que um. Por último, o atributo **"prémio"** que refere aos prémios obtidos por uma determinada série.

- **Temporada**

No caso da entidade **"Temporada"**, existem 3 atributos que a caraterizam. A chave primária é denotada pelo atributo **"IDTemporada"**. Além disso, existem também os atributos **"ano"** e **"número"** que identificam o ano em que a temporada foi lançada e qual o número da mesma, respetivamente.

- **Ator**

Em relação à entidade **"Ator"** representa, tal como o nome indica, um ator de uma determinada série. Este possui os seguintes atributos: **"IDAtor"** que representa o indentificador da entidade, **"nome"**, **"nacionalidade",** "**género"** (masculino ou feminino) e **"dataNascimento"** desse ator.

- **Companhia**

A entidade **"Companhia"** refere-se à empresa que neste momento possui os direitos televisivos referentes a uma ou mais séries. Possui como sua chave primária **"IDCompanhia"** e ainda possui outros atributos como a **"dataFundacao"** que representa a data de fundação da empresa, a **"comissão",** ou seja, o preço a pagar pela Series & Companhia por cada série e o **"nome"** dessa empresa.

- **Criador**

A entidade **"Criador"** destina-se ao armazenamento dos dados relativos a um criador de series. Cada qual é representado por atributos como o **"nome"** do criador, o seu **"género"** (masculino, feminino), **"nacionalidade"** e **"dataNascimento"**. Para ter um identificador único cada criador possui ainda um **"IDCriador".**

- **Serviço**

Para representar um **"Serviço"** na base de dados temos apenas de saber qual o **"preço"** associado ao serviço prestado e qual o seu identificador único de serviço – "**IDServiço"**. Decidimos estabelecer um único tipo de serviço que poderá sofrer variações de preço se tal se justificar no futuro.

- **Episódio**

A entidade **"Episodio"** possui como chave primária **"IDEpisodio"**. Para além disso possui ainda **"legendas"**, sendo possivel ter uma ou mais de diferentes idiomas, **"duração"** que representa o tempo que cada episódio demora, **"número"** do episódio relativamente à temporada, **"avaliação"** fornecida por críticos exteriores à empresa e por fim **"nome"** do episódio em questão.

- **Utilizador**

A entidade **"Utilizador"** tem como identificador único um **"IDUtilizador"**, garantindo assim as propriedades únicas a cada utilizador. Em acréscimo, é representado por atributos simples como o seu **"email"** e o **"nome"** de utilizador, um atributo composto **"morada"** constituído por **"localidade"**, **"rua"**, **"codigoPostal"** e **"pais"** do cliente e um atributo multivalorado **"cartao"** para efeitos de pagamento e faturação.

3.3. Identificação e caracterização dos relacionamentos

No diagrama ER criado foram efetuados diferentes relacionamentos entre entidades.

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade (relacionada)
Criador	N(obrigatório)	Cria	N(obrigatório)	Série
Serviço	1(obrigatório)	Transmite	N(obrigatório)	Série
Utilizador	N(obrigatório)	Subscreve	N(obrigatório)	Serviço
Utilizador	N(obrigatório)	Assiste	N(obrigatório)	Série
Série	1(obrigatório)	Contém	N(obrigatório)	Temporada
Série	N(obrigatório)	Possui	N(obrigatório)	Ator
Companhia	1(obrigatório)	Fornece	N(obrigatório)	Série
Temporada	1(obrigatório)	Contém	N(obrigatório)	Episódio

Tabela 2 - Tabela relacionamentos

- **Relacionamento <u>Criador cria Série</u>**

Requisito de descrição 6: Cada série tem um ou mais criadores.

Descrição: Um criador pode criar várias séries.

Cardinalidade: Criador (1,n) - Série (1,n).

Um criador pode criar várias séries. Várias séries podem ser criadas por um criador.

Atributos: Este relacionamento não possui atributos.

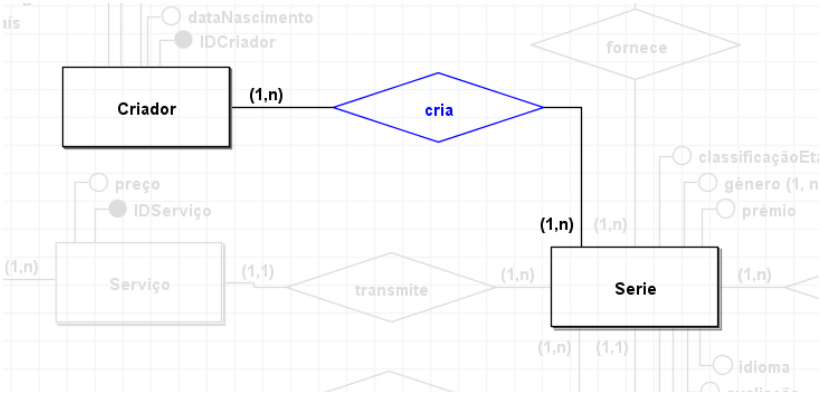

Figura 1 – Ilustração do relacionamento “Criador cria Série”

- **Relacionamento <u>Serviço transmite Série</u>**

Requisito de descrição 8: O serviço consegue transmitir várias séries.

Descrição: Um serviço pode transmitir várias séries.

Cardinalidade: Serviço (1,1) - Série (1,n).

Um serviço pode transmitir várias séries. Uma série só pode ser transmitida num serviço.

Atributos: Este relacionamento não possui atributos.

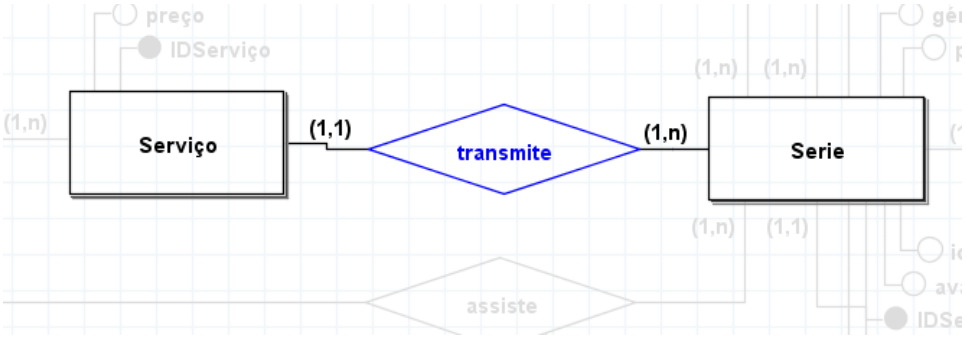

Figura 2 – Ilustração do relacionamento “Serviço transmite Série”

- **Relacionamento <u>Utilizador subscreve Serviço</u>**

Requisito de descrição 2: O utilizador pode subscrever um serviço e irão ficar registadas as datas de início e fim de subscrição assim como o seu preço.

Descrição: Um utilizador pode subscrever um serviço.

Cardinalidade: Utilizador (1,n) - Serviço (1,n).

Um utilizador pode subscrever um ou mais serviços, e um serviço pode ser subscrito por vários utilizadores.

Atributos: Este relacionamento possui 3 atributos: o preço, datas de inicio e fim de subscrição. O “preco” é um atributo do relacionamento dado que depende da data em que é efetuada a subscrição.

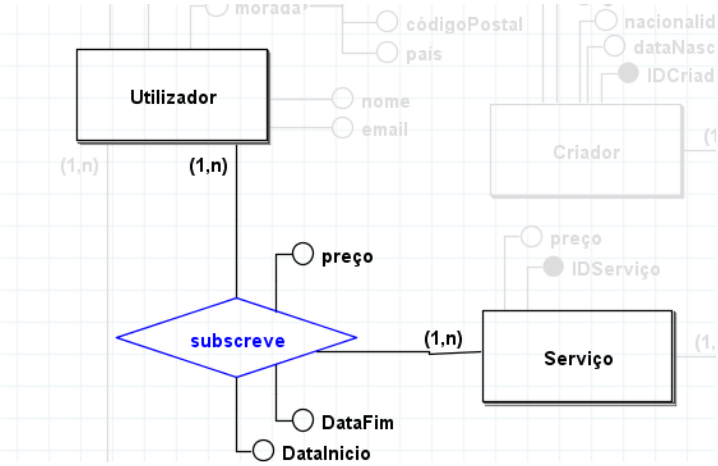

Figura 3 – Ilustração do relacionamento “Utilizador subscreve Serviço”

- **Relacionamento <u>Utilizador assiste Série</u>**

Requisito de descrição 4: O utilizador pode assistir várias séries.

Requisito de descrição 16: Cada utilizador é capaz de marcar uma série como favorita.

Descrição: Um utilizador pode assistir várias séries.

Cardinalidade: Utilizador (1,n) - Série (1,n).

Um utilizador pode assistir a uma ou mais séries. Uma série pode ser assistida por vários utilizadores.

Atributos: O atributo **"favorito"** é um atributo deste relacionamento, que tem como objetivo estabelecer uma preferência por uma determinada série, por parte de um utilizador. Este atributo indica se a série está marcada como favorita ou não.

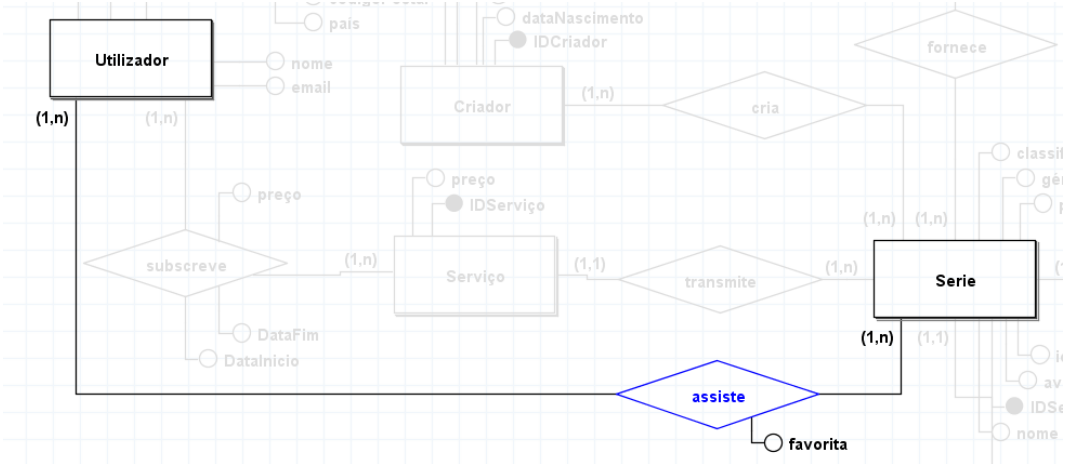

Figura 4 – Ilustração do Relacionamento "Utilizador assiste Série"

- **Relacionamento <u>Série contém Temporada</u>**

Requisito de descrição 5: Cada série contém várias temporadas.

Descrição: As séries contém uma temporada.

Cardinalidade: Série (1,1) – Temporada (1,n)

Uma Série diz respeito a uma ou mais temporadas. Uma temporada só pode fazer parte de uma série.

Atributos: Este relacionamento não possui atributos.

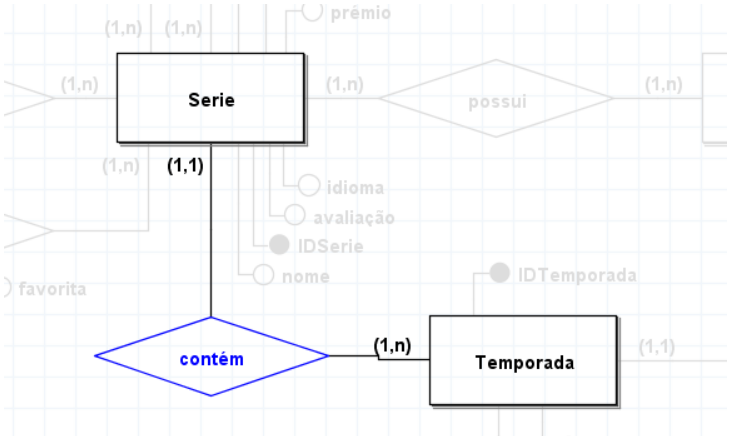

Figura 5 - Ilustração do Relacionamento "Série contém Temporada"

- **Relacionamento <u>Série possui Ator</u>**

Requisito de descrição 9: Uma série possui um elenco principal, constituído por vários atores.

Descrição: Uma série possui um ator no seu elenco.

Cardinalidade: Série (1,n) - Ator (1,n).

Uma série diz respeito a vários atores. Um ator pode estar em várias séries diferentes.

Atributos: Este relacionamento não possui atributos.

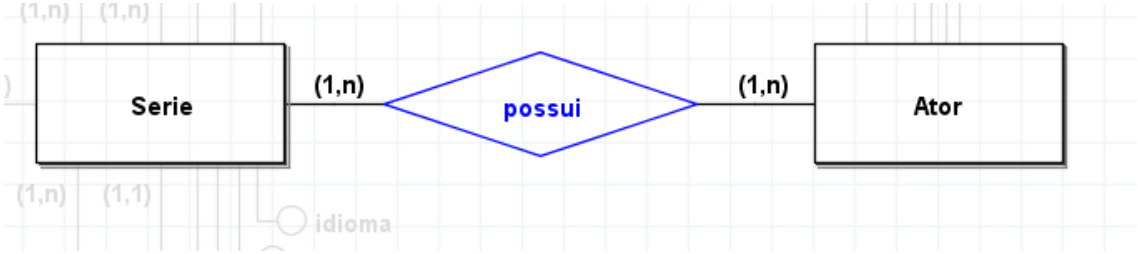

Figura 6 - Ilustração do Relacionamento “Série possui Ator”

- **Relacionamento <u>Companhia fornece Série</u>**

Requisito de descrição 11: Uma série é fornecida por uma companhia.

Descrição: Uma companhia fornece uma série.

Cardinalidade: Companhia (1,1) - Série (1,n).

Uma companhia diz respeito a várias séries. Uma série só diz respeito a uma companhia.

Atributos: Este relacionamento não possui atributos.

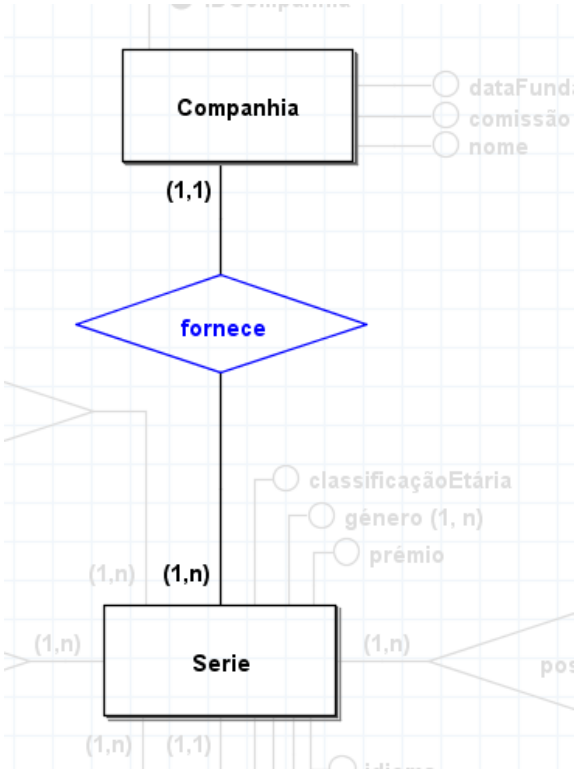

Figura 7 - Ilustração do Relacionamento “Companhia fornece Série”

- **Relacionamento <u>Temporada contém Episódio</u>**

Requisito de descrição 15: Cada temporada tem vários episódios.

Descrição: Uma temporada contém episódios.

Cardinalidade: Temporada (1,1) - Episódio (1,n).

Uma temporada diz respeito a vários episódios. Um episodio diz respeito a uma temporada.

Atributos: Este relacionamento não possui atributos.

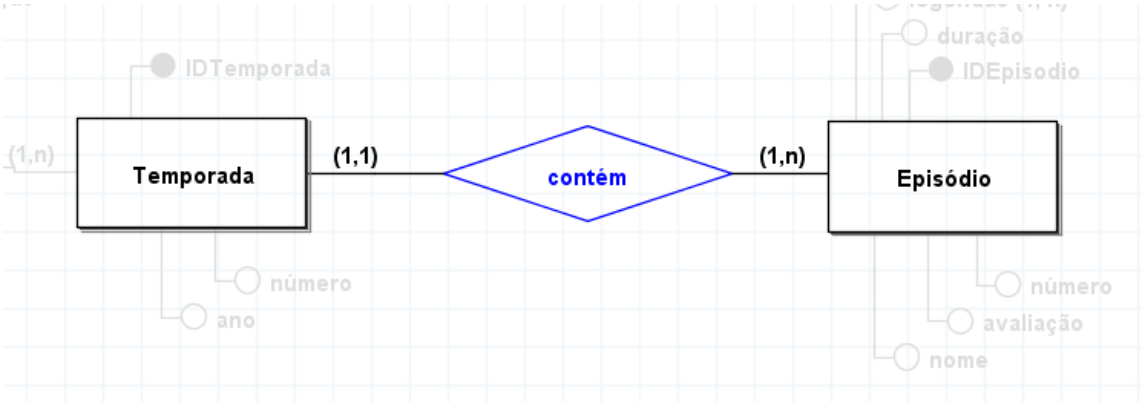

Figura 8 - Ilustração do Relacionamento “Temporada contém Episódio”

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Cada entidade é constituída por vários atributos, que podem ser classificados como simples, derivados, compostos ou multivalor. Alguns atributos são opcionais.

ENTIDADE	ATRIBUTO		TIPO DE DADOS	IDENTIFICADOR	NULO	COMPOSTO	MULTIVALOR	DERIVADO
Utilizador	IdUtilizador		INT	Sim	Não	Não	Não	Não
	Nome		VACHAR(65)	Não	Não	Não	Não	Não
	Endereço	Pais	VACHAR(65)	Não	Não	Sim	Não	Não
		Localidade	VACHAR(60)	Não	Não		Não	Não
		Rua	VACHAR(60)	Não	Não		Não	Não
		Código-Postal	VACHAR(65)	Não	Não		Não	Não
	Email		VACHAR(65)	Não	Não	Não	Não	Não
	Cartão		INT	Não	Não	Não	Sim	Não

Série	IdSérie	INT	Sim	Não	Não	Não	Não
	Nome	VACHAR(65)	Não	Não	Não	Não	Não
	Avaliação	DOUBLE	Não	Não	Não	Não	Não
	Idioma	VACHAR(65)	Não	Não	Não	Não	Não
	Classificação Etária	INT	Não	Não	Não	Não	Não
	Género	VACHAR(65)	Não	Não	Não	Sim	Não
	Prémio	INT	Não	Não	Não	Não	Não
Companhia	IdCompanhia	INT	Sim	Não	Não	Não	Não
	Nome	VACHAR(65)	Não	Não	Não	Não	Não
	Comissão	DOUBLE	Não	Não	Não	Não	Não
	DataFundacao	DATE	Não	Não	Não	Não	Não
Criador	IdCriador	INT	Sim	Não	Não	Não	Não
	Nome	VACHAR(65)	Não	Não	Não	Não	Não
	DataNascimento	DATE	Não	Não	Não	Não	Não
	Nacionalidade	VACHAR(65)	Não	Não	Não	Não	Não
	Género	VACHAR(65)	Não	Não	Não	Não	Não
Ator	IdAtor	INT	Sim	Não	Não	Não	Não
	Nome	VACHAR(65)	Não	Não	Não	Não	Não
	DataNascimento	DATE	Não	Não	Não	Não	Não
	Nacionalidade	VACHAR(65)	Não	Não	Não	Não	Não
	Género	VACHAR(65)	Não	Não	Não	Não	Não
Temporada	IdTemporada	INT	Sim	Não	Não	Não	Não
	Numero	INT	Não	Não	Não	Não	Não
	Ano	INT	Não	Não	Não	Não	Não
Episódio	IdEpisódio	INT	Sim	Não	Não	Não	Não
	Nome	VACHAR(65)	Não	Não	Não	Não	Não
	Avaliacao	DOUBLE	Não	Não	Não	Não	Não
	Numero	INT	Não	Não	Não	Não	Não
	Duracao	TIME	Não	Não	Não	Não	Não
	Legendas	VACHAR(65)	Não	Não	Não	Sim	Não

Serviço	IdServiço	INT	Sim	Não	Não	Não	Não
	Preço	DOUBLE	Não	Não	Não	Não	Não
Subscreve Relacionamento entre utilizador e serviço	DataInicio	DATE	Não	Não	Não	Não	Não
	DataFim	DATE	Não	Não	Não	Não	Não
	Preco	DOUBLE	Não	Não	Não	Não	Não
Assiste Relacionamento entre utilizador e série	Favorita	BOOLEAN	Não	Não	Não	Não	Não

Tabela 3 - Tabela das entidades e atributos

- **Entidade <u>Utilizador</u>**

Requisito de descrição 1: O utilizador pode se registar fornecendo o nome, o cartão para pagamento, a sua morada (localidade, rua, código postal e pais) e o email.

O **"Utilizador"** possui diversos atributos, dos quais o **"IDUtilizador"** trata a chave primária. Nenhum dos atributos pode ser nulo. Possui também um atributo composto **"endereço"** (morada do utilizador) que é constituído pela **"rua", "localidade", "país"** e **"códigoPostal"**. Em adição, possui também um **email** que se trata de um atributo simples e o **cartão** que é composto, dado que pode ter associado mais do que um, de modo a efetuar o pagamento da subscrição.

- **Entidade <u>Série</u>**

Requisito de descrição 7: Uma série tem um nome, avaliação, idioma, classificação etária, vários géneros e pode ter n prémios.

A entidade **"Série"** possui uma chave primária **"IDSérie"**. Além disso, possui um **"nome", "avaliação", "idioma", "classificaçãoEtaria"** e "**premios"**, todos estes classificados como atributos simples. No caso do **"genero"**, este é identificado como multivalor dado que uma série pode estar inserida em vários géneros cinematográficos.

- **Entidade <u>Companhia</u>**

Requisito de descrição 12: Uma companhia contém uma data de fundação, nome e comissão (preço a pagar por série da companhia por parte da **Séries & Companhia**).

A **"Companhia"** possui diversos atributos simples como **"nome", "comissao"** e **"dataFundação".** A sua chave primária é o atributo **"IDCompanhia"**. Todos os atributos devem ser não nulos.

- **Entidade <u>Criador</u>**

Requisito de descrição 17: Um criador é constituído por nome, género, nacionalidade e data de nascimento.

Um **"Criador"** é constituído por atributos simples como **"nome", "nacionalidade", "dataNascimento"** e "**genero"**. O identificador é dado pelo atributo chave **"IDCriador"**.

- **Entidade <u>Ator</u>**

Requisito de descrição 10: Cada ator é constituído pelo seu nome, nacionalidade, género e data de nascimento.

A entidade **"Ator"** é constituída por diversos atributos simples: **"nome", "nacionalidade", "genero"** e **"dataNascimento"**. A chave primária desta entidade é dada por **"IDAtor".**

- **Entidade <u>Temporada</u>**

Requisito de descrição 14: Uma temporada tem um número de temporada e um ano de lançamento.

Na entidade **"Temporada"** estão presentes dois atributos simples: **"numero"** e **"ano"**. A chave primária é dada por **"IDTemporada"**.

- **Entidade <u>Episódio</u>**

Requisito de descrição 13: Um episódio é constituído por um nome, avaliação, número, duração e pode ter vários idiomas de legendas.

A entidade **"Episódio"** possui 6 atributos. O **"IdEpisódio"** é chave primária, mas não é nulo, composto, multivalor nem derivado. Tanto **"nome"**, **"avaliacao"**, **"numero"** e **"duracao"** são atributos simples. Por fim **"legendas"** é um atributo multivalor.

- **Entidade <u>Serviço</u>**

Requisito de descrição 3: O serviço tem um preço associado, que pode variar ao longo do tempo.

A entidade **"Serviço"** contém dois atributos. O **"IDServiço"** é chave primária, no entanto não é nulo, composto, multivalor nem derivado. Quanto ao **"preco"**, este é um atributo simples.

- **Relacionamento <u>Subscreve</u>**

Requisito de descrição 2: O utilizador pode subscrever um serviço e irão ficar registadas as datas de início e fim de subscrição assim como o seu preço.

Nenhum dos atributos desta relação (**"DataInicio"**, **"DataFim", "Preco"**) são identificadores, nulo, composto, multivalor ou derivado.

- **Relacionamento <u>Assiste</u>**

Requisito de descrição 4: O utilizador pode assistir várias séries

Requisito de descrição 16: Cada utilizador é capaz de marcar uma série como favorita.

A relação assiste contém um atributo **"favorita"**. Este atributo não é identificador, nulo, composto, multivalor nem derivado.

3.5. Detalhe ou generalização de entidades

No nosso projeto não foi implementada generalização de entidades. Apesar das entidades **"criador"** e **"ator"** poderem ser generalizadas numa entidade **"pessoa"** (por exemplo) dado os atributos que tem em comum, optamos por não o fazer para manter a distinção entre as entidades.

3.6. Apresentação e explicação do diagrama ER

Até agora foram mencionadas as entidades e os seus atributos numa perspetiva mais singular. Com o intuito de unificar os conceitos supramencionados foi elaborado o diagrama ER. Deste modo, explicitamos na seguinte figura a abordagem unificada de representação entre as diversas entidades e, consequentemente, dos atributos que as constituem.

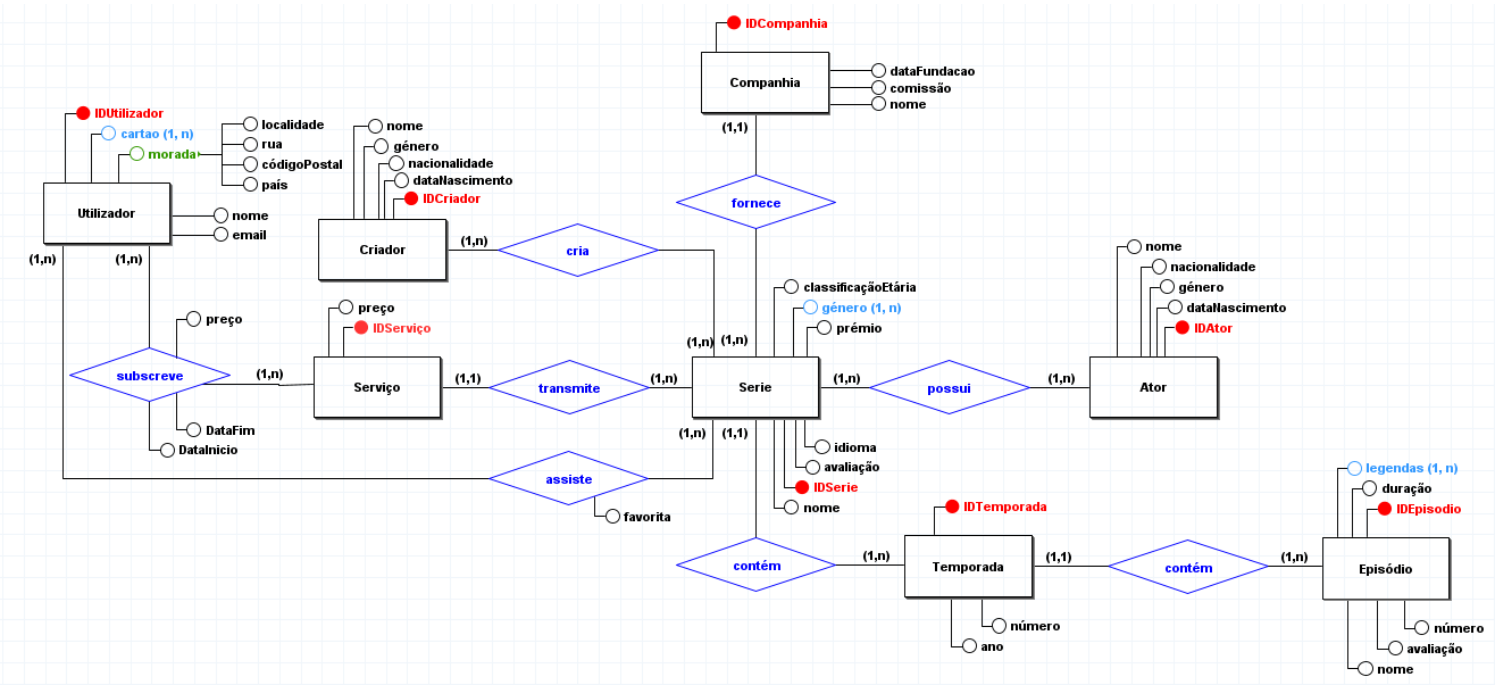

Figura 9 - Ilustração do diagrama ER

O raciocínio implementado pode ser descrito através da leitura do diagrama, da seguinte forma: Um utilizador, ao subscrever um serviço, consegue assistir a várias séries. Existem diversos criadores que as criam, e também várias companhias que as fornecem. Cada série contém várias temporadas que, por sua vez, contém vários episódios. Além disso, cada serie é constituída por vários atores.

3.7. Validação do modelo de dados produzido

Para a validação do modelo, serão analisados os requisitos de descrição. Os restantes serão abordados nas secções posteriores.

1. **O utilizador pode se registar fornecendo o nome, o cartão para pagamento, a sua morada (localidade, rua, código postal e pais) e o email.**

Este requisito é referente à entidade "utilizador" e foi satisfeito da seguinte forma:

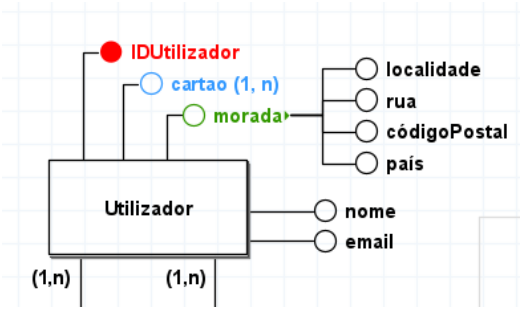

Figura 10 - Entidade "utilizador"

2. **O utilizador pode subscrever um serviço e irão ficar registadas as datas de início e fim de subscrição assim como o seu preço.**

Este requisito foi satisfeito através da relação entre "Utilizador" e "Serviço" ("subscreve"). O preço da subscrição depende da "dataInicio" em que é feita e é válida até à "dataFim", tal como está representado na figura seguinte:

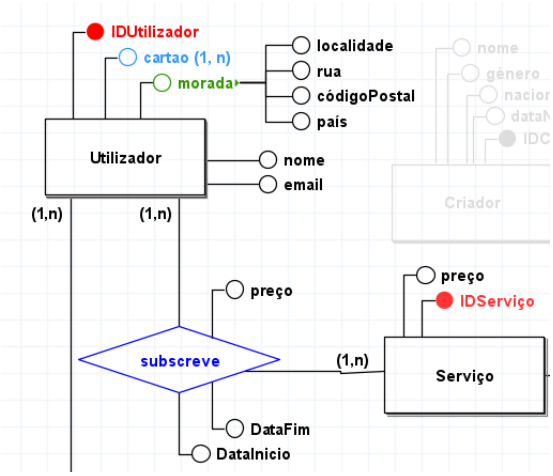

Figura 11 - Relação "Utilizador subscreve Serviço"

3. O serviço tem um preço associado, que pode variar ao longo do tempo.

A seguinte ilustração representa a entidade “Serviço”, que cumpre o requisito:

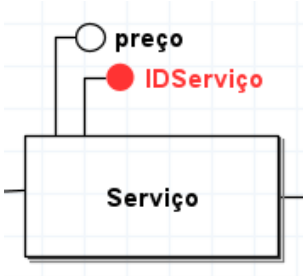

Figura 12 - Entidade “serviço”

4. O utilizador pode assistir várias séries.

Este requisito é satisfeito através da cardinalidade existente entre as entidades “utilizador” e “série”.

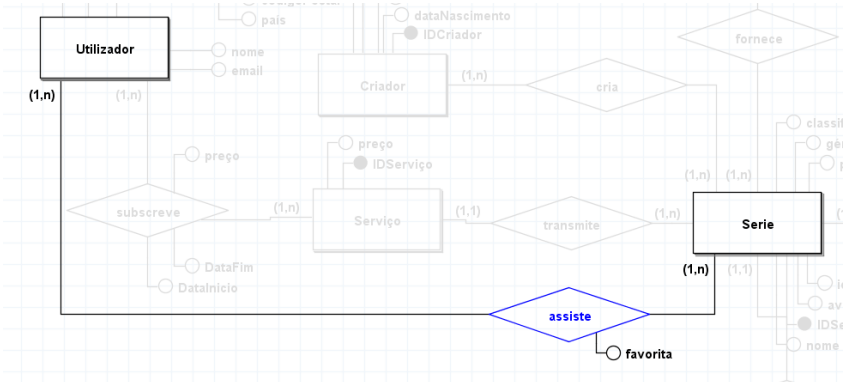

Figura 13 - Relação “Utilizador assiste Série”

5. Cada série contém várias temporadas.

Este requisito é satisfeito através da cardinalidade estabelecida entre as entidades “série” e “temporada”, tal como é visível na seguinte figura:

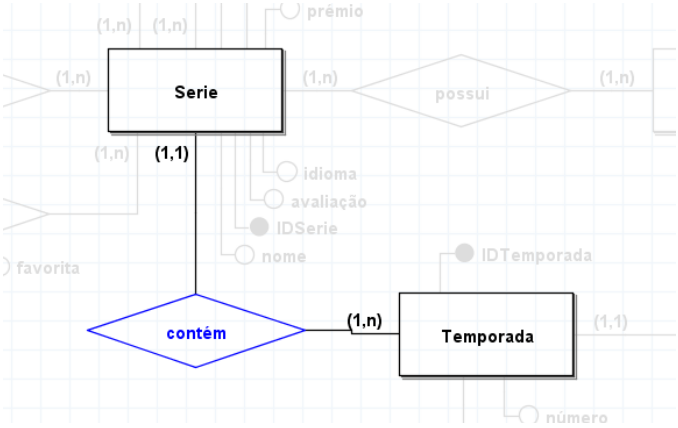

Figura 14 - Relação “Série contém Temporada”

6. **Cada série tem um ou mais criadores.**

Este requisito é satisfeito através da cardinalidade estabelecida entre as entidades “criador” e “série”, tal como é visível na seguinte figura:

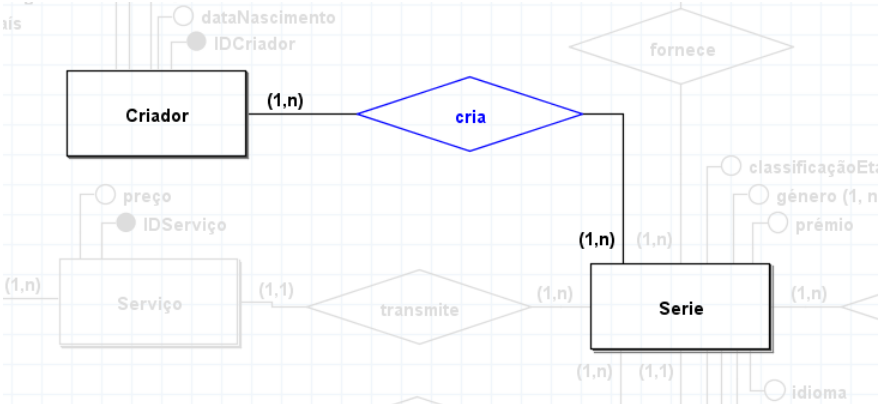

Figura 15 - Relação “Criador cria Série”

7. **Uma série tem um nome, avaliação, idioma, classificação etária, vários géneros e pode ter n prémios.**

O requisito foi satisfeito dados os atributos da entidade “série”, tal como ilustrado na seguinte figura:

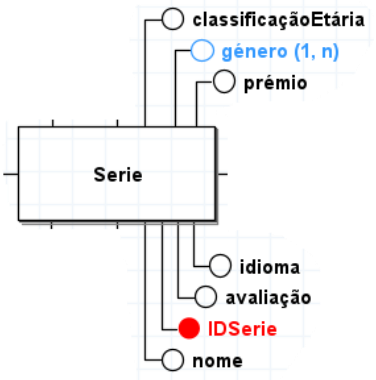

Figura 16 - Entidade “série”

8. **O serviço consegue transmitir várias séries.**

O requisito é satisfeito dada a relação transmite entre “Serviço” e “Série” e a sua cardinalidade:

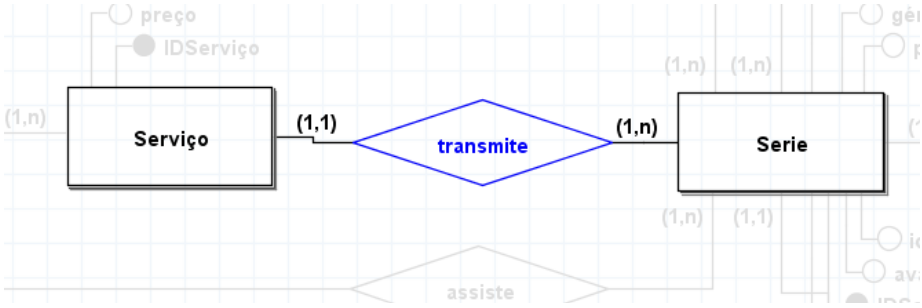

Figura 17 - Relação “Serviço transmite Série”

9. Uma série possui um elenco principal, constituído por vários atores.

Este requisito é satisfeito através da cardinalidade presente entre "Série" e "Ator" (possui), tal como ilustrado no seguinte relacionamento:

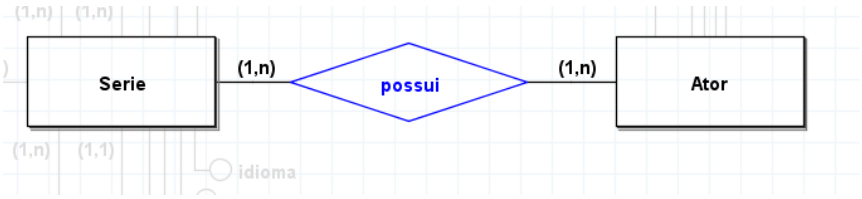

Figura 18 - Relação "Série possui Ator"

10. Cada ator é constituído pelo seu nome, nacionalidade, género e data de nascimento.

Este requisito é validado através da existência dos diversos atributos na entidade respetiva, tal como ilustrado na seguinte figura:

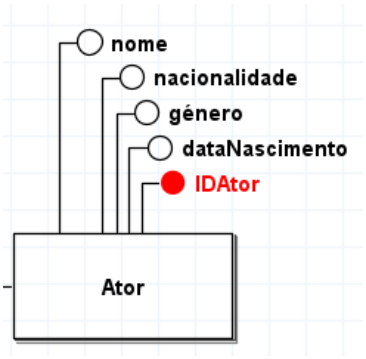

Figura 19 - Entidade "Ator"

11. Uma série é fornecida por uma companhia.

Este requisito é satisfeito através do relacionamento fornece, tal como na figura seguinte:

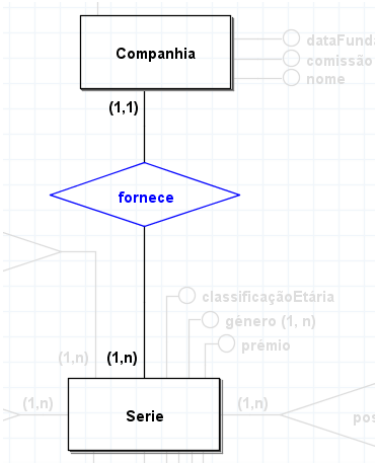

Figura 20 - Relação "Companhia fornece Série"

12. **Uma companhia contém uma data de fundação, nome e comissão (preço a pagar por série da companhia por parte da Séries & Companhia).**

O requisito é cumprido dados os atributos presentes na entidade companhia, tal como ilustrado na figura 21.

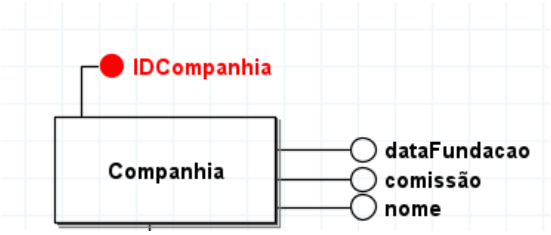

Figura 21 - Entidade “Companhia”

13. **Um episódio é constituído por um nome, avaliação, número, duração e pode ter vários idiomas de legendas.**

O requisito atual é validado através da existência dos atributos necessários à entidade, tal como ilustrado na figura que se segue:

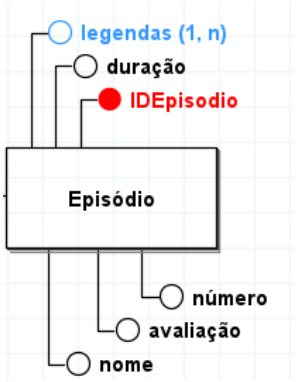

Figura 22 - Entidade “episódio”

14. **Uma temporada tem um número de temporada e um ano de lançamento.**

O requisito é cumprido é validado através da existência dos atributos na seguinte entidade:

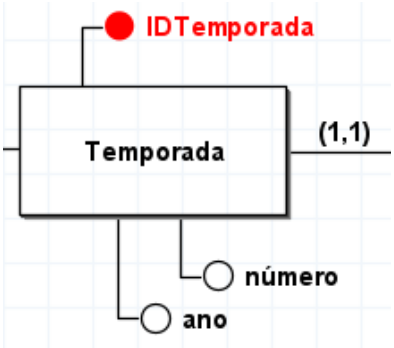

Figura 23 - Entidade “temporada”

15. Cada temporada tem vários episódios.

Este requisito é satisfeito dada a cardinalidade do relacionamento, ilustrada de seguida:

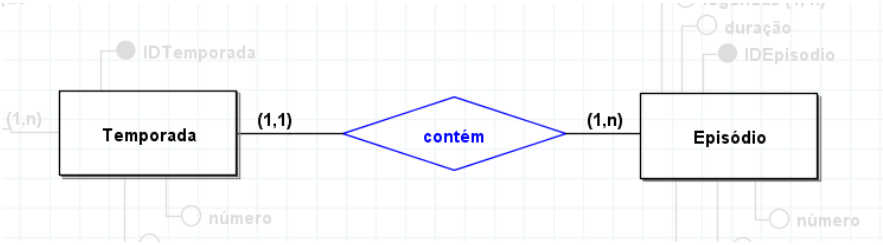

Figura 24 - Relação “Temporada contém Episódio”

16. Cada utilizador é capaz de marcar uma série como favorita.

Este requisito é satisfeito através da existência de um atributo de relação entre as entidades “utilizador” e “série”, tal como descrito na imagem:

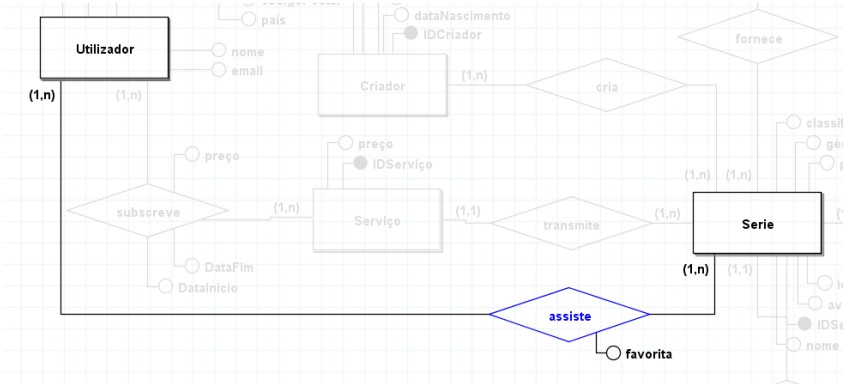

Figura 25 - Atributo “favorito” na relação “assiste”

17. Um criador é constituído por nome, género, nacionalidade e data de nascimento.

Este requisito é satisfeito através dos atributos presente na entidade “Criador”, tal como consta na seguinte figura:

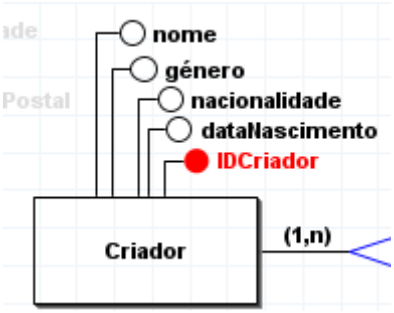

Figura 26 - Entidade “Criador”

4. Modelação lógica

4.1. Construção e validação do modelo de dados lógico

Dada por terminada a conceptualização do nosso problema, com a criação do modelo conceptual acima explicitado, existe a necessidade de converter esse modelo para **modelo lógico,** usando o aplicativo **SQLWorkbench**. Através da criação deste modelo vai ser possivel derivar os relacionamentos entre cada uma das entidades e numa etapa futura povoar a base de dados.

Para a construção do modelo lógico é necessário que para cada entidade se crie uma tabela em que cada entrada representa os atributos definidos no modelo conceptual. Contudo, existem certas regras de transição que devem ser seguidas, tendo em conta os relacionamentos e o tipo de atributos atribuídos no modelo conceptual.

- **Chave Primária**

O primeiro passo é garantir que cada entidade tem uma chave primaria, se não tiver terá de ser criada. Desta forma garantimos a independência entre as entidades. No nosso caso, todas as identidades já possuíam uma chave-primária. Assim fica validado o primeiro passo de transição para o modelo lógico.

- **Relacionamento 1 : N**

Neste tipo de relacionamento, caracterizamos a entidade (1) como sendo pai e a entidade (N) como sendo filha. A chave primária da entidade pai será copiada para a entidade filha como chave estrangeira. Em concreto, esta regra de transição foi aplicada aos relacionamentos da figura 2,5,7,8.

Para os relacionamentos, “companhia **fornece** série” (figura 7) e “serviço **transmite** serie” (figura 2) foi necessário acrescentar á tabela serie as chaves estrangeiras **“companhia**” e **“serviço”** (retângulo vermelho na figura 27), para assim cumprir com a regra acima mencionada.

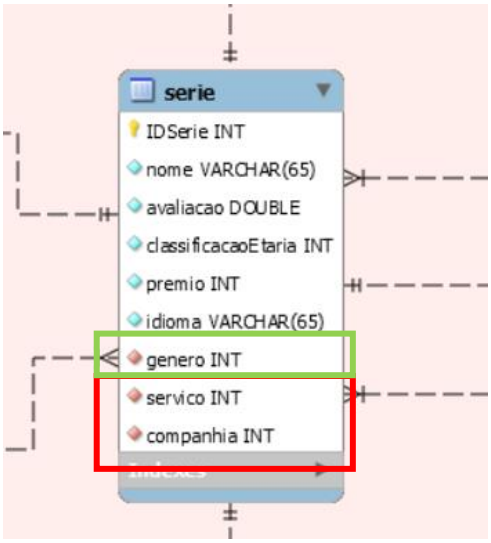

Figura 27 - Tabela da entidade “serie”

Para o relacionamento “serie **contém** temporada” (figura 5) foi necessário acrescentar á tabela “temporada” a chave estrangeira **“serie”** (retângulo vermelho na figura 28), que representa o identificador único da série em questão. Para o relacionamento, “temporada **contém** episodio” (figura 8) foi necessário acrescentar á tabela episodio a chave estrangeira **“temporada”** (retângulo vermelho na figura 29), para identificar a temporada daquele episodio.

Figura 28 - Tabela da entidade “episodio”

Figura 29 - Tabela da entidade “temporada”

- **Relacionamento 1 : 1, 1 : 0, 0 : 0**

A próxima etapa do processo de transição consiste em identificar relacionamentos 1:1, 1:0 ou 0:0. No caso 1:1, temos relações obrigatórias em ambos os lados, podendo optar por juntar ambas numa única entidade conforme se justifique. Em 1:0, uma entidade pode existir e a outra é opcional, devemos então tratar a relação como 1:n (executar segundo passo de transição). Quando temos relacionamentos 0:0, ambas são opcionais. Mediante o contexto temos de escolher uma como pai (1) e a outra como filha (n) e tratar novamente como 1:n (executar segundo passo de transição).

Visto que o nosso modelo conceptual não continha nenhum relacionamento nos formatos descritos podemos avançar este passo de transição.

- **Relacionamento N : N**

O quarto passo das etapas de transição do modelo conceptual para o modelo lógico consiste tratar um relacionamento n:n. Para este efeito, temos de criar uma entidade que inclua as chaves primárias de ambas e, caso aplicável, os atributos da relação.

Assim, para os relacionamentos das figuras 1, 3, 4, 6 criou-se uma tabela com o nome do relacionamento que contém os atributos de relação (para as figuras 4 e 3) e duas chaves estrangeira que representam as chaves primárias de cada uma das entidades do relacionamento. É através destas chaves estrangeiras que conseguimos relacionar as entidades do relacionamento.

Para o relacionamento **"assiste"** (figura 4), visto que este relaciona as entidades "série" e "utilizador", então vai possuir 2 chaves estrangeiras com os identificadores **"serie"** e **"utilizador"**. Como este relacionamento tem um atributo de relação – **"favorito"** - então este também vai estar representado na tabela.

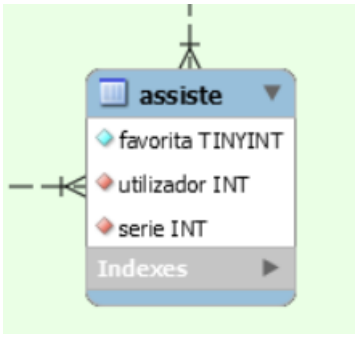

Figura 30 - Tabela do relacionamento "assiste"

Para o relacionamento **"subscreve"** (figura 3), visto que este relaciona as entidades "serviço" e "utilizador", então vai possuir 2 chaves estrangeiras com os identificadores **"servico"** e **"utilizador"**. Como este relacionamento tem os atributos de relação – **"DataInicio", "DataFim", "preco"** - então este também vai estar representado na tabela.

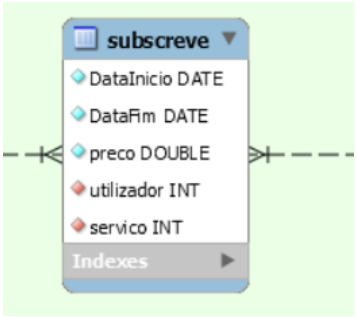

Figura 31 - Tabela do relacionamento "subscreve"

Para o relacionamento **"cria"** (figura 1), visto que este relaciona as entidades "criador" e "série", então vai possuir 2 chaves estrangeiras com os identificadores **"criador"** e **"serie"**. Para o relacionamento **"possui"** (figura 6), visto que este relaciona as entidades "ator" e "série", então vai possuir 2 chaves estrangeiras com os identificadores **"ator"** e **"serie"**.

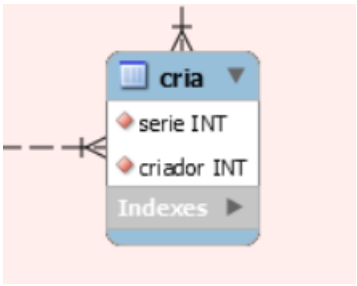

Figura 32 - Tabela do relacionamento "cria"

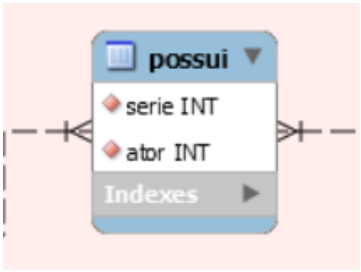

Figura 33 - Tabela do relacionamento "possui"

- **Atributos multivalor**

Finalmente basta verificar a existência de atributos multivalor. Em caso afirmativo, teremos de criar uma entidade (tabela) e tratar como um relacionamento 1: n (executar segundo passo da transição), sendo o atributo multivalor a entidade pai (1) e a alternativa a entidade filha (n).

Objetivamente, isto traduz-se em transformar em tabelas os atributos de multivalor: **"género"** (que pertence à entidade "Série"), **"legendas"** (que pertence à entidade "Episódio") e **"cartão"** (que pertence à entidade "Utilizador"). Podemos ver nas figuras 34, 36 e 37 as tabelas obtidas no modelo lógico. Para além disso é necessário que, nas tabelas das entidades que possuem estes atributos, tenham uma chave estrangeira que represente o identificador do respetivo atributo. Assim a tabela **"série"** tem de ter uma chave estrangeira que represente o género (retângulo verde na figura 27), a tabela **"episódio"** tem uma chave estrangeira que representa as legendas (retângulo verde na figura 28) e a tabela **"utilizador"** tem uma chave estrangeira que representa o cartão (retângulo verde na figura 35).

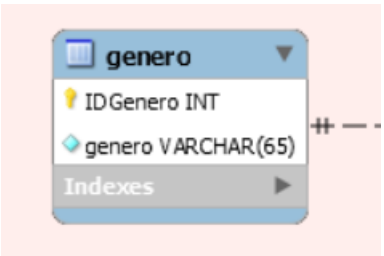

Figura 34 - Tabela do atributo "genero"

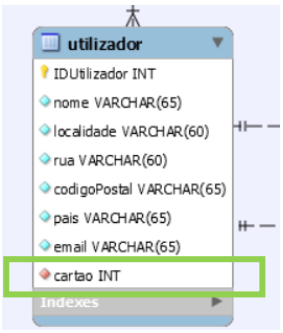

Figura 35 - Tabela da entidade "utilizador"

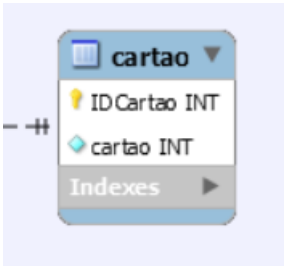

Figura 37 - Tabela do atributo "legenda"

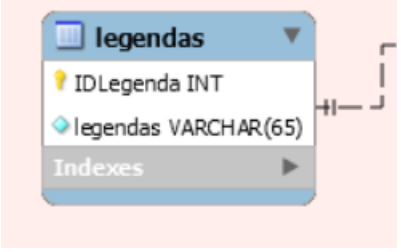

Figura 36 - Tabela do atributo "cartao"

4.2. Desenho do modelo lógico

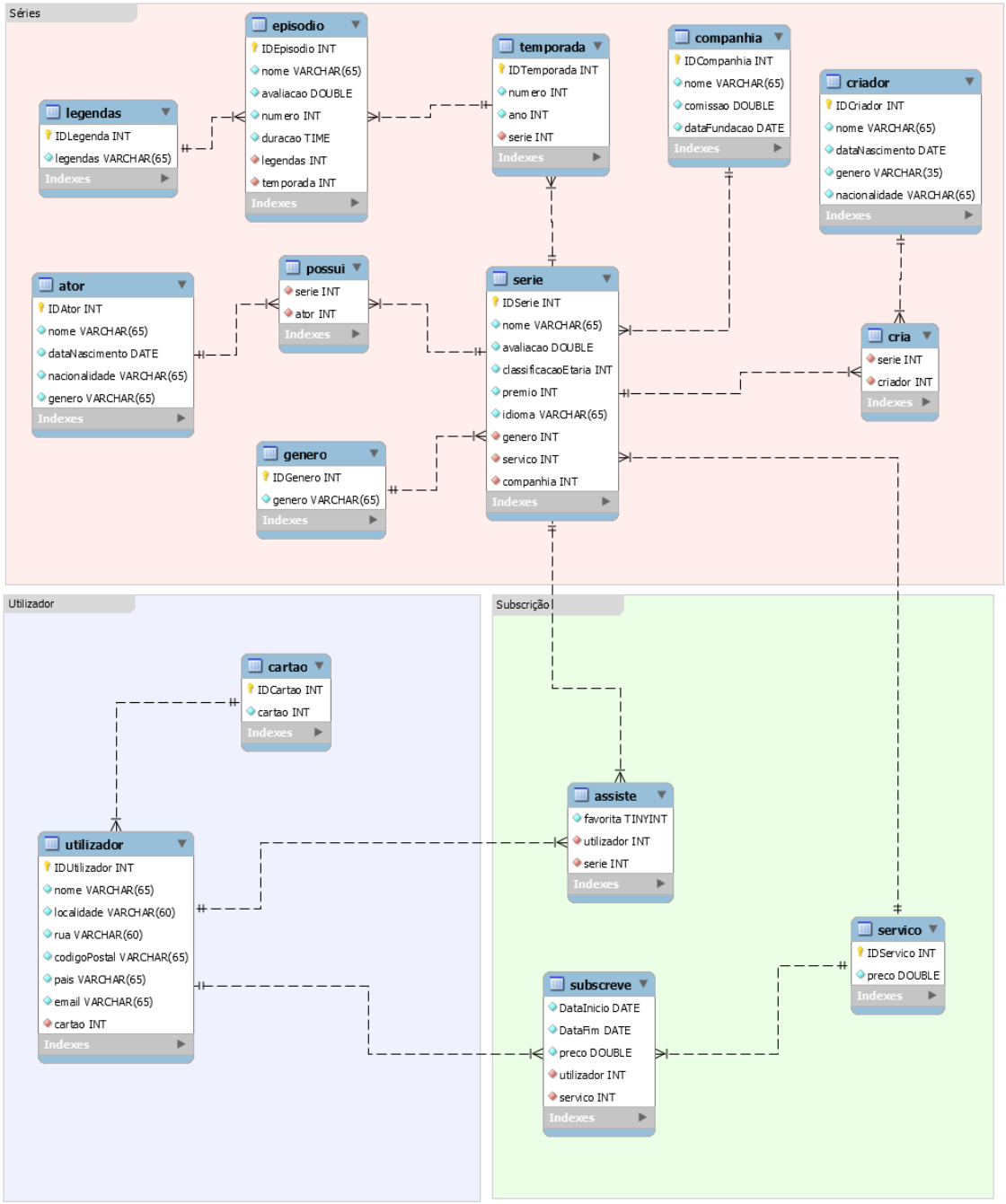

Figura 38 - Modelo Lógico

As camadas – Utilizador, Subscrição, Serie – estão representadas a cores para facilitar a organização do modelo, contudo estas camadas não representam diferentes vistas do modelo lógico. Funcionam exclusivamente para organização do modelo.

4.3. Validação do modelo através da normalização

De forma a validar o modelo é necessário recorrer à normalização. Existem 3 formas normais de validação por este método:

- **Primeira Forma Normal (1FN):**

Nesta forma normal é necessário localizar todos os atributos multivalorados e torná-los atributos simples. Para isso, é necessário remover esses atributos da tabela da entidade correspondente, dando origem a uma nova tabela para o atributo multivalor. Esta tabela é constituída pelo próprio atributo e um identificador. Existirá depois uma relação chave primária – chave estrangeira entre esta nova tabela e a tabela da entidade que lhe deu origem. No nosso caso, foi necessário realizar este passo nos atributos **"género"** (da entidade **"série"**), **"cartão"** (da entidade **"utilizador"**) e **"legendas"** (da entidade **"episódio"**).

- **Segunda Forma Normal (2FN):**

Para o modelo respeitar a 2FN, é necessário que respeite a 1FN. A segunda forma normal não permite que atributos sejam totalmente independentes da chave primária. No caso destes atributos existirem, deverão ser retirados da tabela da entidade e serem colocados numa nova tabela. No nosso sistema, cada atributo não chave depende da totalidade da chave, logo respeita a segunda forma normal.

- **Terceira Forma Normal (3FN):**

De forma a que o modelo esteja na 3FN, é necessário que respeite a 2FN. Para estar na terceira forma normal é necessário que nenhum atributo não chave dependa funcionalmente de um outro atributo que não seja o identificador. Caso se verifique dependência funcional entre atributos não chave, é necessário removê-los da tabela e colocá-los numa nova. Estas duas tabelas irão ficar conectadas através da par chave primária – chave estrangeira. Dado que não existe nenhuma dependência funcional entre atributos não chave, concluímos que o nosso modelo está na 3FN.

4.4. Validação do modelo com interrogações do utilizador

A validação do modelo passa por elaborar a álgebra relacional das interrogações do utilizador.

1. **O sistema deverá ser capaz de identificar qual o top N das séries mais vistas.**

Começamos por fazer a junção das tabelas “assiste” e “serie”, através dos atributos “serie” e “IDSerie”, respetivamente. De seguida, foram agrupados por nome de série e efetuada a contagem dos seus utilizadores. Posteriormente foram ordenados por ordem decrescente e estabelecido um limite de 3 séries (por exemplo). Por fim, foram selecionadas as colunas do “nome” e do “nrUtilizadores” para apresentar os resultados.

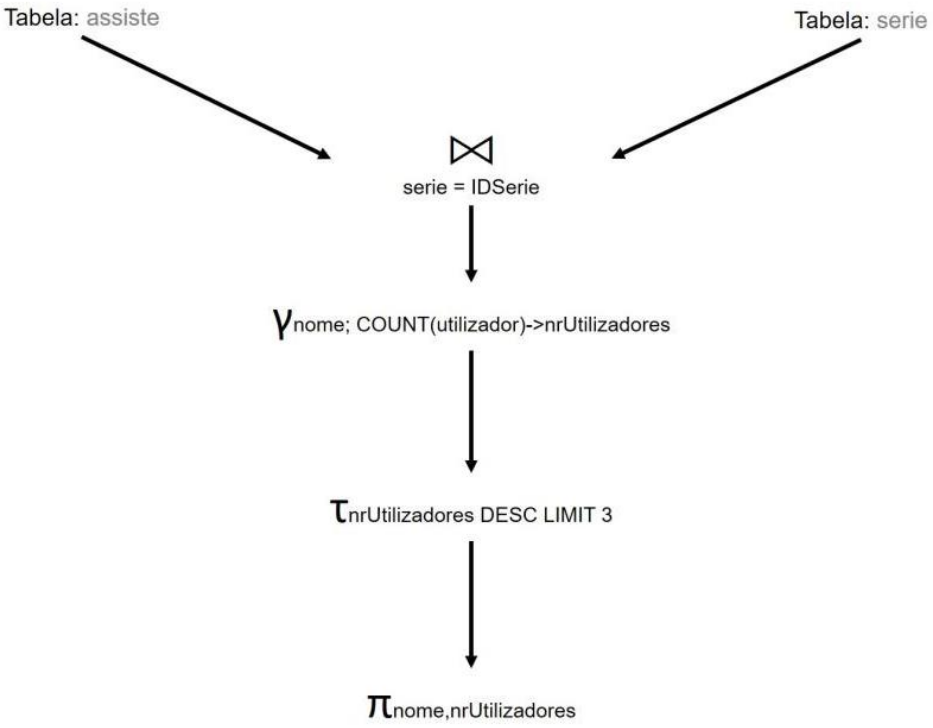

Figura 39 - Álgebra relacional *query* 1

2. **O sistema deverá ser capaz de identificar qual o top N de series com melhor classificação.**

Primeiramente, acedemos à tabela “serie” e ordenamos por ordem decrescente do atributo “avaliação” e limitamos a tabela resultante a 1 (por exemplo). Para finalizar, projetamos as colunas “nome” e “avaliação” para obter o resultado pretendido.

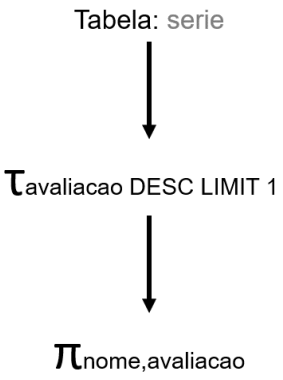

Figura 40 - Álgebra relacional *query* 2

3. **O sistema deverá ser capaz de identificar quais as companhias que forneceram mais séries.**

Inicialmente, recorreu-se à junção das tabelas “companhia” e “série”, através dos atributos “IDCompanhia” e “companhia”, respetivamente. De seguida, efetuou-se um agrupamento por nome de companhia e contagem do número de séries respetivo. Após agrupar consoante o pretendido, ordenou-se a tabela por ordem decrescente da contagem efetuada. Por último, selecionaram-se as colunas “nome” e “nrSeries”.

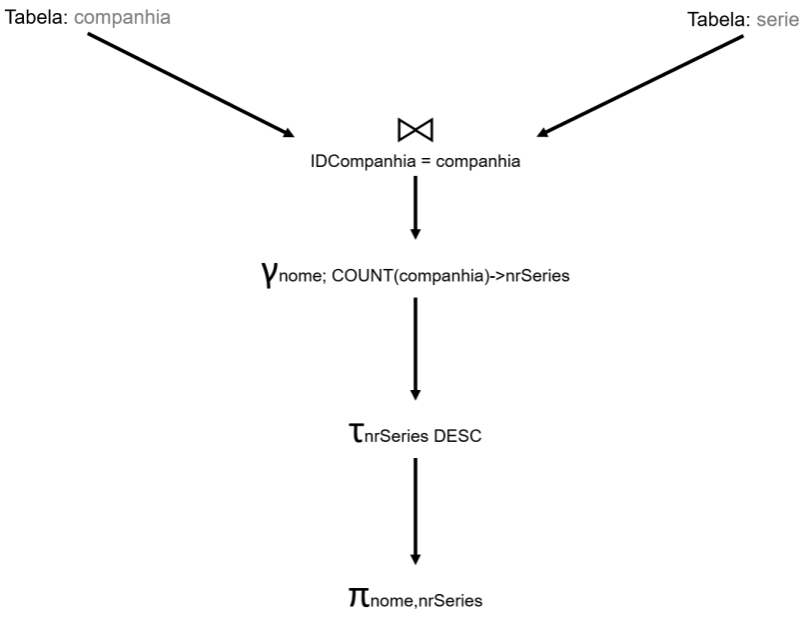

Figura 41 - Álgebra relacional *query* 3

4. **O sistema deverá ser capaz de apresentar todas as séries e o seu número de prémios.**

Começou-se por aceder à tabela “serie” e ordenar por ordem decrescente de “prémio”. Por último, efetuou-se uma projeção das colunas “nome” e “premio”, de modo a obter o resultado pretendido.

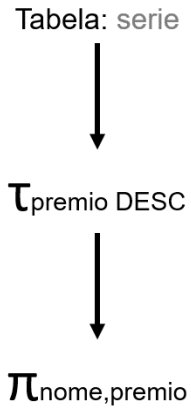

Figura 42 - Álgebra relacional *query* 4

5. **O sistema deverá ser capaz de identificar qual é top N dos utilizadores que assistem mais séries.**

Inicialmente recorremos ao agrupamento das tabelas “assiste” e “utilizador”, a partir dos atributos “utilizador” e “IDUtilizador”, respetivamente. De seguida, efetuou se o agrupamento por nome de utilizador e contagem das séries. Posteriormente, foi elaborada uma ordenação decrescente desta contagem, e, de modo a obter o resultado pretendido, foram projetadas as colunas “nome” e “nrSeries”.

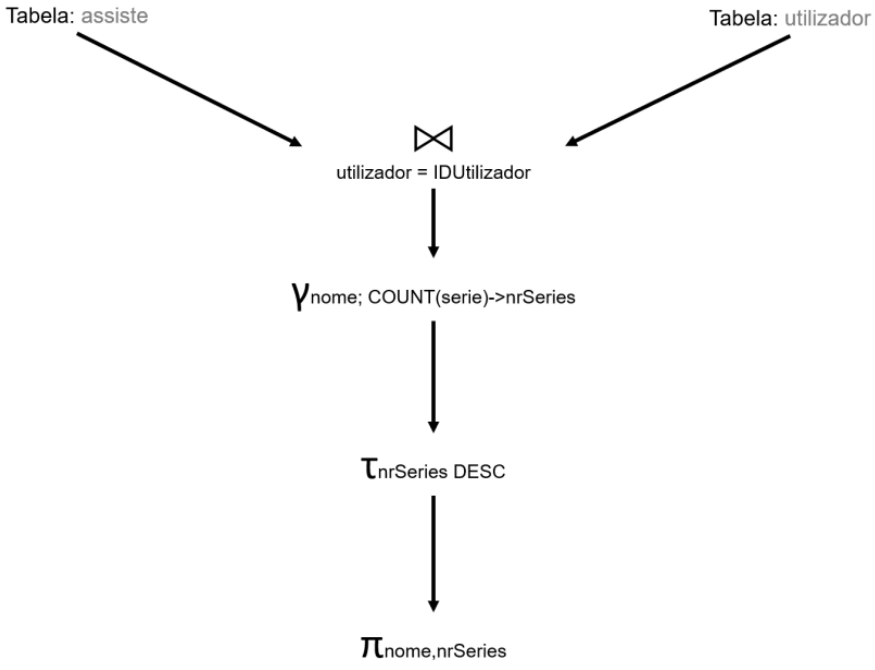

Figura 43 - Álgebra relacional *query* 5

6. **O sistema deverá ser capaz de identificar séries de um determinado idioma.**

Primeiramente, recorremos à tabela “série” e efetuamos uma seleção das séries com o idioma desejado (a título de exemplo, utilizamos o idioma inglês). Por último, efetuamos uma projeção das colunas “nome” e “idioma” da série.

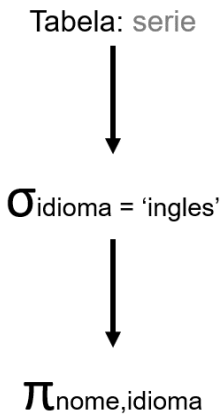

Figura 44 - Álgebra relacional *query* 6

7. **O sistema deverá ser capaz de identificar séries cujos episódios possuem legendas de um determinado idioma.**

Primeiramente, recorreu-se à junção das tabelas “temporada” e “série”, através dos atributos “série” e “IDSerie”, respetivamente. De seguida, juntou-se a esta mesma a tabela “série”, através dos atributos “IDTemporada” e “temporada”, respetivamente. Posteriormente, à tabela resultante juntou-se a tabela “legendas”, através dos atributos “legendas” e “IDLegendas”, respetivamente. Na tabela resultante, efetuou-se uma seleção de legendas pelo idioma pretendido (no exemplo, português) e de seguida um agrupamento pelo nome de série e fez-se a contagem do número de vezes que a legenda pretendida apareceu na série. Por último, foi elaborada uma projeção das colunas nome de série, a contagem efetuada anteriormente e o idioma da legenda.

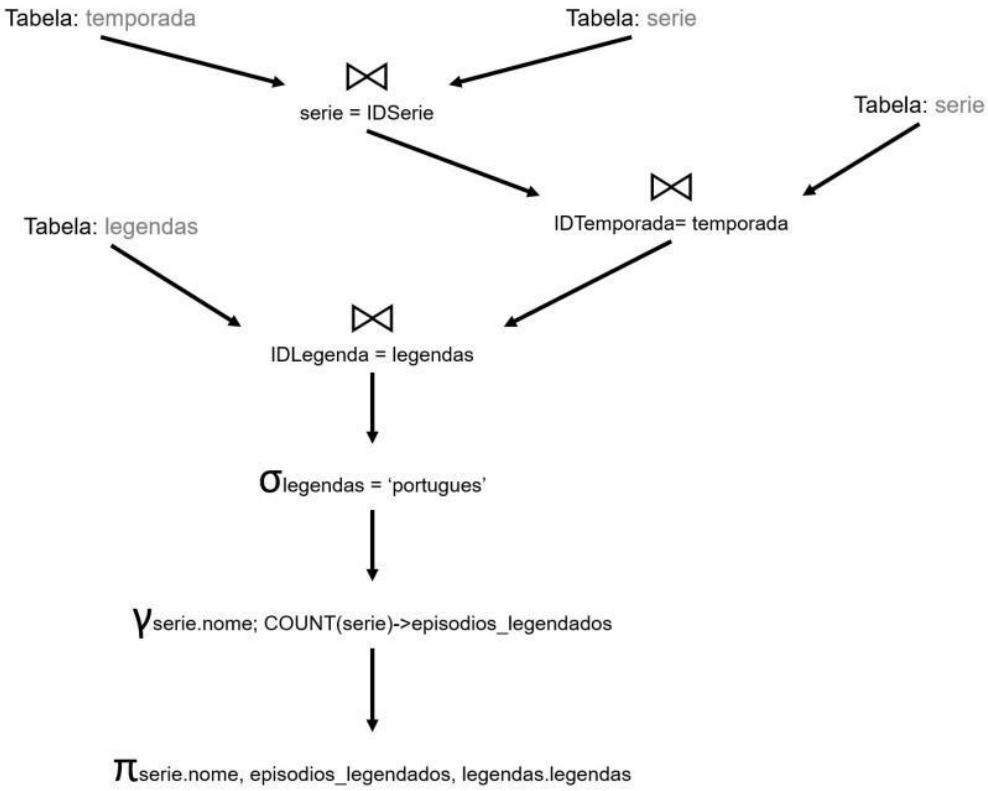

Figura 45 - Álgebra relacional *query* 7

8. **O sistema deverá ser capaz de identificar quais as séries apropriadas para uma determinada faixa etária.**

Começou-se por recorrer à tabela "série" e efetuar uma seleção das séries cuja classificação etária é inferior ao valor desejado (no exemplo, 16). Por último, efetuou-se uma projeção do nome das séries e da respetiva classificação etária.

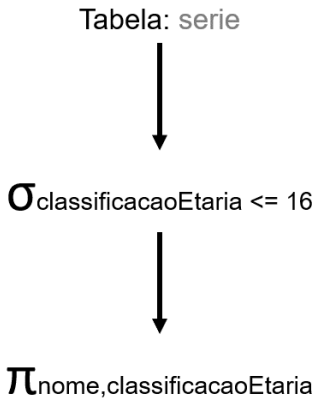

Figura 46 - Álgebra relacional *query* 8

9. **O sistema deverá ser capaz de identificar quais as séries favoritas de um utilizador.**

Inicialmente, recorreu-se à junção das tabelas “assiste” e “serie”, através dos atributos “serie” e “IDSerie”, respetivamente. De seguida, juntou-se a tabela resultante com a tabela “utilizador”, através dos atributos “utilizador” e “IDUtilizador”, respetivamente. Posteriormente, efetuou-se uma seleção na tabela das séries favoritas (favorito = 1) e do respetivo utilizador (neste caso, o utilizador com ID 4). Por último, projetou-se o nome de utilizador e o nome das séries.

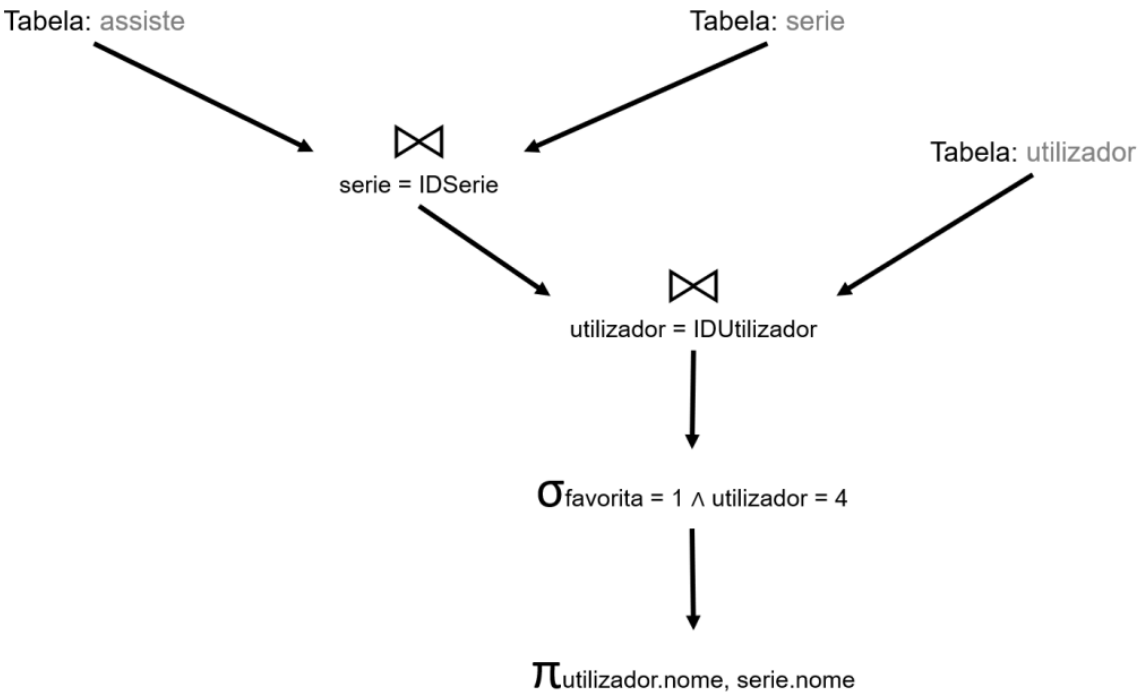

Figura 47 - Álgebra relacional *query* 9

10. O sistema deverá ser capaz de identificar qual o número de utilizadores com conta ativa.

Inicialmente, efetuou-se o agrupamento das tabelas “subscreve” e “utilizador”, através dos atributos “utilizador” e “IDUtilizador”, respetivamente. De seguida, efetuou-se uma seleção das linhas cuja data de fim é superior à data atual. Por último, foi efetuada uma projeção com o nome de utilizador e datas de inicio e fim de subscrição.

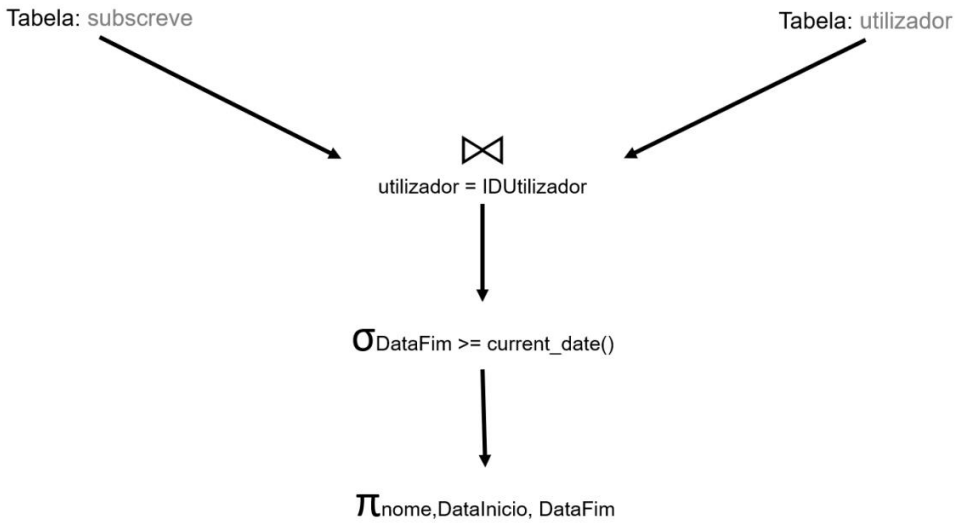

Figura 48 - Álgebra relacional *query* 10

11. O sistema deverá ser capaz de identificar qual a faturação total.

Inicialmente, foi selecionada a tabela “subscreve” e agrupada pela soma da sua faturação total. Por último, foi selecionada essa mesma faturação.

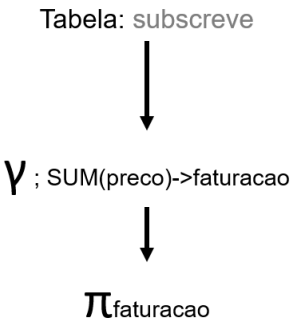

Figura 49 - Álgebra relacional *query* 11

4.5. Revisão do modelo lógico produzido

No final desta fase, ficou concluído o modelo lógico. Como tal, é necessário reunir a equipa de modo a validar o mesmo e poder prosseguir para a implementação física. O objetivo da reunião é apresentar e discutir o modelo de modo a combinar diferentes perspetivas e chegar a um consenso de implementação que satisfaça as necessidades da empresa.

Verificamos que todas as entidades do modelo conceptual estão presentes no modelo lógico, assim como os respetivos atributos e também que o seu tipo de dados permite armazenar a informação necessária.

Deste modo, após obter a validação necessária e consenso entre todos os membros da equipa, foi nos possível continuar com o projeto para a implementação física.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

O sistema de gestão de bases de dados assegura aos seus utilizadores serviços de descrição, manutenção, exploração e controlo de dados.

Assim, de modo a ir de encontro aos serviços pretendidos, optamos por utilizar o *MySQL*. Este foi também o *software* com o qual tivemos contacto nas aulas práticas e estávamos mais familiarizados.

À parte disso, trata-se de um *software* que apresenta um bom desempenho, uma vez que é leve e a sua performance não é muito afetada pelo aumento da dimensão da base de dados. Trata-se também de uma ferramenta gratuita, que se torna acessível para todos, e possui muitos utilizadores pelo mundo, o que facilita a resolução de problemas que possam aparecer ao longo do projeto.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

Após a conclusão do esquema lógico, foi elaborado o esquema físico correspondente. Visto que construímos o modelo lógico no *MySQL Workbench*, conseguimos tirar partido das suas funcionalidades, nomeadamente o *Forward Engineering* que este disponibiliza. Esta opção permite gerar código SQL automaticamente a partir do modelo lógico. Deste modo, conseguimos construir a base de dados na linguagem SQL, de forma simples e eficaz.

5.3. Tradução das interrogações do utilizador para SQL

Os requisitos de exploração podem ser traduzidos em instruções SQL. Deste modo, para cada requisito, abaixo apresenta-se o código SQL e os procedimentos criados para os representar.

1. **Identificar qual o top N de séries mais vistas.**

```
-- Query 1) Identificar qual o top N das série mais vista.
DELIMITER $$
DROP PROCEDURE IF EXISTS topNSeriesMaisVistas;
CREATE PROCEDURE topNSeriesMaisVistas (IN topN INT)
BEGIN
SELECT S.nome, count(A.utilizador) AS nrUtilizadores
    FROM assiste AS A
        JOIN serie AS S ON S.IDSerie = A.serie
        GROUP BY A.serie
            ORDER BY nrUtilizadores DESC
            LIMIT topN;
END $$
```

Figura 50 - *Query* 1 *MySQL*

2. **Identificar a qual o top N de série com a melhor classificação.**

```
-- Query 2) Identificar qual o top N de series com melhor classificação.
DELIMITER $$
DROP PROCEDURE IF EXISTS topNSerieMelhorClassificada;
CREATE PROCEDURE topNSerieMelhorClassificada(IN topN INT)
BEGIN
SELECT nome, avaliacao
    FROM serie
        ORDER BY avaliacao DESC
            LIMIT topN;
END $$
```

Figura 51 - *Query* 2 *MySQL*

3. Identificar quais as companhias que forneceram mais séries.

```
-- Query 3) O sistema deverá ser capaz de identificar quais as companhias que forneceram mais séries.
DELIMITER $$
DROP PROCEDURE IF EXISTS MaioresCompanhias;
CREATE PROCEDURE MaioresCompanhias()
BEGIN
SELECT C.nome, count(S.companhia) AS nrSeries
    FROM serie AS S
        JOIN companhia AS C ON C.IDCompanhia = S.companhia
            GROUP BY S.companhia
            ORDER BY nrSeries DESC ;
END $$
```

Figura 52 - *Query* 3 *MySQL*

4. Identificar todas as series e o seu número de prémios.

```
-- Query 4) O sistema deverá ser capaz de apresentar todas as series e o seu numero de premios.
DELIMITER $$
DROP PROCEDURE IF EXISTS TopSeriesPremios;
CREATE PROCEDURE TopSeriesPremios()
BEGIN
SELECT S.nome, S.premio
    FROM serie AS S
        ORDER BY S.premio DESC;
END $$
```

Figura 53 - *Query* 4 *MySQL*

5. Identificar qual o top N dos utilizadores que assistem mais series.

```
-- Query 5) O sistema deverá ser capaz de identificar qual é top N dos utilizadores que vê mais séries.
DELIMITER $$
DROP PROCEDURE IF EXISTS NrSeriesPorUtilizador;
CREATE PROCEDURE NrSeriesPorUtilizador(IN topN INT)
BEGIN
SELECT U.nome , count(A.serie) AS nrSeries
 FROM assiste AS A
    JOIN utilizador AS U ON U.IDUtilizador = A.utilizador
        GROUP BY A.utilizador
            ORDER BY nrSeries DESC
                LIMIT topN;
END $$
```

Figura 54 - *Query* 5 *MySQL*

6. **Identificar as séries de um determinado idioma.**

```
-- Query 6) O sistema deverá ser capaz de identificar as séries de um determinado idioma.
DELIMITER $$
DROP PROCEDURE IF EXISTS ProcuraSeriesNacionalidade;
CREATE PROCEDURE ProcuraSeriesNacionalidade(IN idioma VARCHAR(65))
BEGIN
SELECT S.idioma, S.nome
    FROM serie AS S
        WHERE S.idioma = idioma; -- exemplo: ingles, italiano, espanhol
END $$
```

Figura 55 - *Query* 6 *MySQL*

7. **Identificar as séries cujos episódios possuem legendas de um determinado idioma.**

```
-- Query 7) O sistema deverá ser capaz de identificar séries cujos episódios possuem legendas de um determinado idioma.
DELIMITER $$
DROP PROCEDURE IF EXISTS ProcuraSeriesTraduzidas;
CREATE PROCEDURE ProcuraSeriesTraduzidas(IN legenda VARCHAR(65))
BEGIN
SELECT S.nome, COUNT(serie) AS Episodios_Legendados, L.legendas
    FROM serie AS S
        JOIN temporada AS T ON T.serie = S.IDSerie
        JOIN episodio AS E ON T.IDTemporada = E.temporada
        JOIN legendas AS L ON L.IDLegenda = E.legendas
            WHERE L.legendas = legenda  -- exemplo: ingles, portugues, espanhol
                GROUP BY S.nome;
```

Figura 56 - *Query* 7 *MySQL*

8. **Identificar quais as series apropriadas para uma determinada faixa etária.**

```
-- Query 8) O sistema deverá ser capaz de identificar quais as séries apropriadas para uma determinada faixa etária.
DELIMITER $$
DROP PROCEDURE IF EXISTS ConsultaSeriesIdade;
CREATE PROCEDURE ConsultaSeriesIdade(IN idade INT)
BEGIN
SELECT S.nome , S.classificacaoEtaria
    FROM serie AS S
        WHERE S.classificacaoEtaria <= idade; -- exemplo: 12, 14, 16, 18
END $$
```

Figura 57 - *Query* 8 *MySQL*

9. Identificar quais as séries favoritas de um determinado utilizador.

```
-- Query 9) O sistema deverá ser capaz de identificar quais as séries favoritas de um utilizador.
DELIMITER $$
DROP PROCEDURE IF EXISTS SeriesFavoritasUtilizador;
CREATE PROCEDURE SeriesFavoritasUtilizador(IN idUtilizador INT)
BEGIN
SELECT U.nome, S.nome AS SeriesFavoritas
    FROM assiste AS A
            JOIN serie AS S ON A.serie = S.IDSerie
            JOIN utilizador AS U ON A.utilizador = U.IDUtilizador
            WHERE A.favorita = 1
                    && A.utilizador = idUtilizador; -- exemplo: 1,..,10
END $$
```

Figura 58 - *Query* 9 *MySQL*

10. Identificar qual o número de utilizadores que possuem conta ativa.

```
-- Query 10) O sistema deverá ser capaz de identificar qual o numero de utilizadores com conta ativa.
DELIMITER $$
DROP PROCEDURE IF EXISTS UtilizadoresAtivos;
CREATE PROCEDURE UtilizadoresAtivos()
BEGIN
SELECT U.nome, S.DataInicio, S.DataFim
    FROM subscreve AS S
    JOIN utilizador AS U ON U.IDUtilizador = S.utilizador
        WHERE S.DataFim >= current_date();
END $$
```

Figura 59 - *Query* 10 *MySQL*

11. Identificar qual a faturação total

```
-- Query 11) O sistema deverá ser capaz de identificar qual a faturação total.
DELIMITER $$
DROP PROCEDURE IF EXISTS FaturacaoTotal;
CREATE PROCEDURE FaturacaoTotal()
BEGIN
SELECT SUM(S.preco) AS Faturação
    FROM subscreve AS S
        GROUP BY S.servico;
END $$
```

Figura 60 - *Query* 11 *MySQL*

5.4. Escolha, definição e caracterização de índices em SQL

A utilização de índices permite melhorar o desempenho da base de dados. O *MySQL*, por defeito, cria índices relativos às chaves primárias e estrangeiras de cada tabela.

No nosso trabalho, dada a reduzida dimensão da base de dados e quantidade de registos, a utilização de índices não introduzirá grande diferença em termos de desempenho. No entanto, caso a base de dados possuísse uma escala maior, ou na eventualidade de aumentar a dimensão da BD, será útil ter certos atributos indexados para facilitar a consulta dos mesmos, e deste modo melhorar a *performance* da base de dados.

Assim, a título de exemplo, poderíamos criar um índice único relativo ao atributo “email” da tabela “utilizador”, dado que cada *email* é único e não se repete. Deste modo, a consulta a este atributo desta entidade será mais rápida e eficiente.

```
CREATE UNIQUE INDEX idxEmail
ON utilizador (email) ;
```

Figura 61 - Índice "idxEmail"

5.5. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Para o cálculo do espaço necessário para cada entrada na tabela foi utilizado o capítulo 11.7 do *MySQL 8.0 Reference Manual*, de forma a verificar o espaço necessário para cada atributo.

TIPO DE DADOS	TAMANHO (em bytes)
DOUBLE	8
VARCHAR(m)	L+1, com L o tamanho da *string*
DATE	3
TIME	3
INT	4
TINYINT	1

Tabela 4 - Tamanho dos tipos de dados usados

É necessário ter em consideração que os tamanhos de certos atributos vão variando de acordo com os valores introduzidos, por exemplo, para o nome do utilizador, apesar de este conseguir tem um nome de 65 caracteres, o espaço armazenado neste campo (assim como

todos os campos que envolvam *strings*) depende do número de caracteres introduzidos para o nome. Assim, o espaço utilizado por cada entrada vai diferindo.

De forma a melhor estimar o espaço usado, foi construída a tabela abaixo onde estão apresentados o espaço em média utilizado, isto é, para cada valor das *strings* qual é o valor máximo normalmente utilizado e o espaço máximo que pode ser utilizado:

TABELA	ESPAÇO MÉDIA DE UMA ENTRADA (em bytes)	ESPAÇO MÁXIMO DE UMA ENTRADA (em bytes)
Utilizador	≈ 117	388
Cartao	8	8
Serie	≈ 64	162
Genero	≈ 15	69
Episodio	≈ 83	92
Legendas	≈ 15	69
Temporada	16	16
Companhia	≈ 26	80
Ator	≈ 60	202
Possui	8	8
Criador	≈ 60	172
Cria	8	8
Servico	12	12
Subscreve	22	22
Assiste	9	9

Tabela 5 - Tamanho por entrada de cada em tabela

Multiplicando os valores da tabela de cima pela quantidade de entradas presentes em cada tabela, obteve-se o seguinte quadro:

TABELA	ESPAÇO MÉDIA DA POPULAÇÃO (em bytes)	ESPAÇO MÁXIMO DA POPULAÇÃO (em bytes)
Utilizador	≈ 1170	3880
Cartao	80	80
Serie	≈ 512	1296
Genero	≈ 75	345
Episodio	≈ 9379	10396
Legendas	≈ 45	207
Temporada	208	208
Companhia	≈ 130	400

Ator	≈ 1080	3636
Possui	144	144
Criador	≈ 540	1548
Cria	72	72
Servico	12	12
Subscreve	220	220
Assiste	405	405
∑	14072	22849

Tabela 6 - Tamanho da população de cada tabela

Contudo, se avaliarmos o tamanho da BD segundo a ferramenta *Schema Inspector* do *MySQL workbench* observamos que os valores obtidos são diferentes. Na realidade ambos estão corretos. No sentido em que a tabela acima mencionada representa apenas o espaço ocupado pelo povoamento atual.

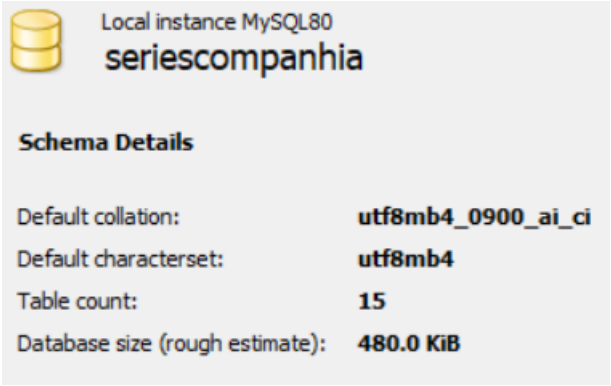

Figura 62 - Tamanho da BD

Anualmente, estimamos que o número de utilizadores aumente para os 50 000, o que vai levar ao aumento da ocupação da base de dados em 5 850 000 bytes (≈5.9 MB) relativamente à tabela “utilizador”, 400 000 bytes relativos ao cartão e 1 100 000 bytes (≈1.1 MB) relativos á tabela “subscreve”. Para além destes aumentos fixos, a tabela “assiste” também vai aumentar a sua ocupação, contudo este aumento vai depender do número de séries que cada utilizador vê.

Prevê-se que por ano sejam acrescentadas ao sistema mais 24 séries. Este incremento leva ao aumento direto do espaço ao ocupado pela tabela series em 1536 bytes. Em média, cada série possui 3 temporadas e 15 episódios, 5 atores (com 5 novas entradas na tabela “possui”) e 1 criador (com 1 nova entrada na tabela “cria”). Além disso estima-se que a cada 12 séries deve aumentar 1 género. Estes aumentos refletem-se num espaço ocupado de cerca de 1 716 bytes por série. No total, o espaço ocupado seria de cerca de 1716 * 24 +1536 = 42 720 bytes.

Se fosse decidido criar 1 novo serviço por ano, o espaço da base de dados aumentaria 12 bytes. Para além disso, se por ano fizéssemos 2 novos contratos com novas companhias refletir-se-ia num aumento de 52 bytes.

Assim, por ano, estima-se que a base de dados tenha um aumento de armazenamento de cerca de 7 392 784 bytes ≈ 7.4 MB.

5.6. Definição e caracterização das vistas de utilização em SQL

Uma vista trata-se de uma tabela lógica criada em *runtime* e pode ser vista como um elemento de dados. Assim, ao criar vistas conseguimos ter a visualização de uma *query* e consultar os seus dados como se fosse uma tabela. Além disso, uma vista também tem como função delimitar os dados de uma tabela. No nosso caso, este último não será o propósito de criação das vistas, dado que trabalhamos na BD como administradores. No entanto, é útil a utilização das mesmas no caso de *queries* mais complicadas.

Deste modo, a título de exemplo, decidimos criar uma vista com a tabela resultante da *query* 10:

```
CREATE VIEW vwUtilizadoresAtivos AS
SELECT U.nome, S.DataInicio, S.DataFim
    FROM subscreve AS S
    JOIN utilizador AS U ON U.IDUtilizador = S.utilizador
        WHERE S.DataFim >= current_date();
```

Figura 63 - Vista da *query* 10

E ainda, para aceder ao conteúdo desta vista, poderíamos utilizar o seguinte comando:

```
SELECT *
    FROM vwutilizadoresativos;
```

Figura 64 - Instrução para aceder ao conteúdo da vista

	nome	DataInicio	DataFim
▶	Anastasia Batista	2019-12-27	2020-12-27
	Deolinda Pereira Cardoso	2020-02-28	2021-02-28
	Duarte Macedo da Rocha	2020-03-20	2021-03-20
	Fernando Botelho Lopes	2020-04-10	2021-04-10
	Ana Lu Bretice Renoir	2020-04-17	2021-04-17
	John Marco Holmes	2020-05-13	2021-05-13

Figura 65 - Resultado da execução da vista da *query* 10

Decidimos construir também uma vista referente à *query* 3, tal como de seguida se apresenta:

```
CREATE VIEW vwmaiorescompanhias AS
SELECT C.nome, count(S.companhia) AS nrSeries
    FROM serie AS S
        JOIN companhia AS C ON C.IDCompanhia = S.companhia
            GROUP BY S.companhia
            ORDER BY nrSeries DESC ;
```

Figura 66 – Vista da *query* 3

5.7. Revisão do sistema implementado

Dados por concluída esta fase do projeto, procedeu-se a uma nova e última reunião na series & companhia, com o objetivo de validar todas as etapas feitas até então e possivelmente dar por concluído o projeto.

Primeiramente, começamos por verificar se todas as interrogações foram implementadas de forma a cumprir com os requisitos propostos anteriormente. De seguida, foi discutido e validado a implementação dos índices de forma a melhorar a performance de pesquisa da base de dados. Além disso, também foi analisado o espaço ocupado atualmente pela BD e estimou-se o aumento do armazenamento desta após um ano, seguindo as espectativas de crescimento da empresa. Finalmente, avaliamos a implementação de algumas vistas para facilitar o acesso a esses dados ou então para poderem ser usados outros intervenientes.

Ao fim da reunião, conclui-se que toda a implementação estava correta e seguia os modelos previstos, o que permitiu avançar para a instalação do sistema de gestão de base de dados.

6. Conclusões e Trabalho Futuro

Com a criação e implementação da base de dados para plataforma Séries & Companhia, concluímos que o desenvolvimento de BD para esta aplicação é a uma mais valia pois tira proveito de uma tecnologia de gestão, exploração e armazenamento de dados, de forma a garantir a longevidade da plataforma facilitando a manutenção dos dados armazenadas e ter a capacidade de conseguir suportar um aumento de utilizadores devido á situação pandémica atual.

As fases seguidas no desenvolvimento da BD foram derivadas do método de conceção de base de dados apresentado no livro *Database Systems, A Practical Approach to Design, Implementation, and Management.* Este método visa decompor o caso de estudo em diversas fases, tendo como prioridade cumprir com a totalidade dos requisitos, de forma a ir de encontro aos critérios estabelecidos pela aplicação.

A BD desenvolvida cumpre com os requisitos de exploração e descrição propostos pela plataforma Séries & Companhia o que faz com que seja funcional e consistente. A base de dados implementada é dotada de grande escalabilidade dado que possui margem para crescimento. Assim, será possível implementar novas funcionalidades e acrescentar novos dados e registos, tai como a subscrição a novos e diversos serviços premium que assegura a expansão da plataforma melhorando e diversificando a experiência de cada utilizador. Ao longo da sua implementação obedecemos às regras de normalização com o intuito de melhorar a estruturação dos modelos criados.

Em virtude da minimização da redundância dos dados tomamos especial atenção à inserção dos dados, por forma a evitar o armazenamento dos dados em lugares distintos. Para além de uma boa estrutura, primamos na apresentação de dados íntegros, evitando o uso de atributos vazios e conteúdo falacioso. De modo a conferir maior segurança na base de dados foi adotado um sistema de backup durante as horas de menor tráfego, uma vez que é necessário desligar os servidores para realizar o mesmo.

Em alternativa, poderíamos ter criado uma entidade “pessoa” que englobasse as entidades ator e criador, pois estas possuem atributos idênticos. Além disso, poderíamos ter complementado a nossa base de dados com uma entidade faturação, por exemplo, em que gerava uma fatura por cada serviço pago.

Num olhar para o futuro, sentimos que existe um enorme potencial ainda por explorar. A expansão da empresa passará certamente pelo mercado de *streaming watch party*, ou seja, ter a oportunidade de ver séries em modo de transmissão partilhada com outras pessoas em

residências diferentes. Este novo mercado levará à criação de novos serviços e funcionalidades a acrescentar não só na plataforma, mas também na base de dados criada.

Dado por concluído o projeto, consideramos que na era digital com constantes progressos tecnológicos, os conceitos de gestão de base de dados revelam-se fulcrais para enfrentar as promessas do futuro.

Referências

Chinchlikar, C., 2020. *The billion-dollar opportunity the Indian OTT industry is missing.* [Online]
Available at: https://yourstory.com/2020/05/billion-dollar-opportunity-indian-ott-industry-missing
[Acedido em Novembro 23 2020].

Compos, J., 2020. *Plataformas de Streaming: conheça as alternativas à Netflix em Portugal.* [Online]
Available at: https://selectra.pt/tv-net-voz/servicos/streaming
[Acedido em 22 Novembro 2020].

MySQL, 2020. *MySQL 8.0 Reference Manual.* [Online]
Available at: https://dev.mysql.com/doc/refman/8.0/en/
[Acedido em 1 Dezembro 2020].

Pallotta, F., 2020. *Streaming services are missing this feature if they want to dominate the pandemic era.* [Online]
Available at: https://edition.cnn.com/2020/06/17/media/streaming-watch-party-hulu-netflix/index.html
[Acedido em 22 Novembro 2020].

Lista de Siglas e Acrónimos

BD Base de Dados

ER *Entity–Relationship*

SQL *Structured Query Language*

Anexos

I. Anexo 1 – *Script* de criação da base de dados

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET                        @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET                                              @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE
RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydbassiste
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema seriescompanhia
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema seriescompanhia
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `seriescompanhia`;
CREATE  SCHEMA  IF  NOT  EXISTS  `seriescompanhia`  DEFAULT  CHARACTER  SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `seriescompanhia` ;

-- -----------------------------------------------------
-- Table `seriescompanhia`.`cartao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`cartao` (
  `IDCartao` INT NOT NULL,
  `cartao` INT NOT NULL,
  PRIMARY KEY (`IDCartao`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`utilizador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`utilizador` (
  `IDUtilizador` INT NOT NULL,
  `nome` VARCHAR(65) NOT NULL,
  `localidade` VARCHAR(60) NOT NULL,
  `rua` VARCHAR(60) NOT NULL,
  `codigoPostal` VARCHAR(65) NOT NULL,
  `pais` VARCHAR(65) NOT NULL,
  `email` VARCHAR(65) NOT NULL,
  `cartao` INT NOT NULL,
  PRIMARY KEY (`IDUtilizador`),
  INDEX `FK_Utilizador_2` (`cartao` ASC) VISIBLE,
  CONSTRAINT `FK_Utilizador_2`
    FOREIGN KEY (`cartao`)
    REFERENCES `seriescompanhia`.`cartao` (`IDCartao`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`genero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`genero` (
  `IDGenero` INT NOT NULL,
  `genero` VARCHAR(65) NOT NULL,
  PRIMARY KEY (`IDGenero`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`servico`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`servico` (
```

```
  `IDServico` INT NOT NULL,
  `preco` DOUBLE NOT NULL,
  PRIMARY KEY (`IDServico`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`companhia`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`companhia` (
  `IDCompanhia` INT NOT NULL,
  `nome` VARCHAR(65) NOT NULL,
  `comissao` DOUBLE NOT NULL,
  `dataFundacao` DATE NOT NULL,
  PRIMARY KEY (`IDCompanhia`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`serie`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`serie` (
  `IDSerie` INT NOT NULL,
  `nome` VARCHAR(65) NOT NULL,
  `avaliacao` DOUBLE NOT NULL,
  `classificacaoEtaria` INT NOT NULL,
  `premio` INT NOT NULL,
  `idioma` VARCHAR(65) NOT NULL,
  `genero` INT NOT NULL,
  `servico` INT NOT NULL,
  `companhia` INT NOT NULL,
  PRIMARY KEY (`IDSerie`),
  INDEX `FK_Serie_2` (`genero` ASC) VISIBLE,
  INDEX `FK_Serie_3` (`servico` ASC) VISIBLE,
  INDEX `FK_Serie_4` (`companhia` ASC) VISIBLE,
  CONSTRAINT `FK_Serie_2`
    FOREIGN KEY (`genero`)
```

```
    REFERENCES `seriescompanhia`.`genero` (`IDGenero`),
  CONSTRAINT `FK_Serie_3`
    FOREIGN KEY (`servico`)
    REFERENCES `seriescompanhia`.`servico` (`IDServico`)
    ON DELETE RESTRICT,
  CONSTRAINT `FK_Serie_4`
    FOREIGN KEY (`companhia`)
    REFERENCES `seriescompanhia`.`companhia` (`IDCompanhia`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`assiste`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`assiste` (
  `favorita` TINYINT NOT NULL,
  `utilizador` INT NOT NULL,
  `serie` INT NOT NULL,
  INDEX `FK_assiste_1` (`utilizador` ASC) VISIBLE,
  INDEX `FK_assiste_2` (`serie` ASC) VISIBLE,
  CONSTRAINT `FK_assiste_1`
    FOREIGN KEY (`utilizador`)
    REFERENCES `seriescompanhia`.`utilizador` (`IDUtilizador`)
    ON DELETE RESTRICT,
  CONSTRAINT `FK_assiste_2`
    FOREIGN KEY (`serie`)
    REFERENCES `seriescompanhia`.`serie` (`IDSerie`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`ator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`ator` (
  `IDAtor` INT NOT NULL,
```

```
  `nome` VARCHAR(65) NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `nacionalidade` VARCHAR(65) NOT NULL,
  `genero` VARCHAR(65) NOT NULL,
  PRIMARY KEY (`IDAtor`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`criador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`criador` (
  `IDCriador` INT NOT NULL,
  `nome` VARCHAR(65) NOT NULL,
  `dataNascimento` DATE NOT NULL,
  `genero` VARCHAR(35) NOT NULL,
  `nacionalidade` VARCHAR(65) NOT NULL,
  PRIMARY KEY (`IDCriador`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`cria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`cria` (
  `serie` INT NOT NULL,
  `criador` INT NOT NULL,
  INDEX `FK_cria_1` (`serie` ASC) VISIBLE,
  INDEX `FK_cria_2` (`criador` ASC) VISIBLE,
  CONSTRAINT `FK_cria_1`
    FOREIGN KEY (`serie`)
    REFERENCES `seriescompanhia`.`serie` (`IDSerie`)
    ON DELETE RESTRICT,
  CONSTRAINT `FK_cria_2`
    FOREIGN KEY (`criador`)
    REFERENCES `seriescompanhia`.`criador` (`IDCriador`)
    ON DELETE RESTRICT)
```

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`legendas`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`legendas` (
  `IDLegenda` INT NOT NULL,
  `legendas` VARCHAR(65) NOT NULL,
  PRIMARY KEY (`IDLegenda`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`temporada`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`temporada` (
  `IDTemporada` INT NOT NULL,
  `numero` INT NOT NULL,
  `ano` INT NOT NULL,
  `serie` INT NOT NULL,
  PRIMARY KEY (`IDTemporada`),
  INDEX `FK_Temporada_2` (`serie` ASC) VISIBLE,
  CONSTRAINT `FK_Temporada_2`
    FOREIGN KEY (`serie`)
    REFERENCES `seriescompanhia`.`serie` (`IDSerie`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`episodio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`episodio` (
  `IDEpisodio` INT NOT NULL,
```

```
  `nome` VARCHAR(65) NOT NULL,
  `avaliacao` DOUBLE NOT NULL,
  `numero` INT NOT NULL,
  `duracao` TIME NOT NULL,
  `legendas` INT NOT NULL,
  `temporada` INT NOT NULL,
  PRIMARY KEY (`IDEpisodio`),
  INDEX `FK_Episodio_2` (`legendas` ASC) VISIBLE,
  INDEX `FK_Episodio_3` (`temporada` ASC) VISIBLE,
  CONSTRAINT `FK_Episodio_2`
    FOREIGN KEY (`legendas`)
    REFERENCES `seriescompanhia`.`legendas` (`IDLegenda`),
  CONSTRAINT `FK_Episodio_3`
    FOREIGN KEY (`temporada`)
    REFERENCES `seriescompanhia`.`temporada` (`IDTemporada`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `seriescompanhia`.`possui`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`possui` (
  `serie` INT NOT NULL,
  `ator` INT NOT NULL,
  INDEX `FK_possui_1` (`serie` ASC) VISIBLE,
  INDEX `FK_possui_2` (`ator` ASC) VISIBLE,
  CONSTRAINT `FK_possui_1`
    FOREIGN KEY (`serie`)
    REFERENCES `seriescompanhia`.`serie` (`IDSerie`)
    ON DELETE RESTRICT,
  CONSTRAINT `FK_possui_2`
    FOREIGN KEY (`ator`)
    REFERENCES `seriescompanhia`.`ator` (`IDAtor`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----------------------------------------------------
-- Table `seriescompanhia`.`subscreve`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `seriescompanhia`.`subscreve` (
  `DataInicio` DATE NOT NULL,
  `DataFim` DATE NOT NULL,
  `preco` DOUBLE NOT NULL,
  `utilizador` INT NOT NULL,
  `servico` INT NOT NULL,
  INDEX `FK_subscreve_1` (`utilizador` ASC) VISIBLE,
  INDEX `FK_subscreve_2` (`servico` ASC) VISIBLE,
  CONSTRAINT `FK_subscreve_1`
    FOREIGN KEY (`utilizador`)
    REFERENCES `seriescompanhia`.`utilizador` (`IDUtilizador`)
    ON DELETE RESTRICT,
  CONSTRAINT `FK_subscreve_2`
    FOREIGN KEY (`servico`)
    REFERENCES `seriescompanhia`.`servico` (`IDServico`)
    ON DELETE RESTRICT)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

II. Anexo 2 - *Script* de povoamento

```
-- Esquema: "seriescompanhia"
USE `seriescompanhia` ;

--
-- Permissão para fazer operações de remoção de dados.
SET SQL_SAFE_UPDATES = 0;

--

-- Povamento da tabela "cartao"
INSERT INTO cartao
      (IDCartao, cartao)
    VALUES
    (1, 12335),
    (2, 65546),
      (3, 33333),
    (4, 44444),
      (5, 55555),
    (6, 66666),
      (7, 77777),
    (8, 88888),

    (9, 99999),
    (10, 10101)
    ;

-- Povoamento da tabela "Utilizador"
INSERT INTO utilizador
      (IDUtilizador, nome, localidade, rua, codigoPostal, pais, email,
cartao)
      VALUES
            (1, 'João da Costa e Campos', 'Braga', 'Rua das Adegas
Felizes, 12, 1ª Cave', '4710-035', 'Portugal', 'jcc@acacia.pt',1),
```

```
            (2, 'Josefina Vivida da Paz', 'Lisboa', 'Av dos Castros
Reais,      122,      3°      Andar',      '4520-980',      'Portugal',
'josefina@acacia.pt',2),
        (3, 'Joaquim da Silva', 'Porto', 'Rua das Pancadas, 3, 4
Andar°', '4200-135', 'Portugal', 'jds@acacia.pt',3),
            (4, 'António Ramalhos', 'Aveiro', 'Av das Rosas, 122, 1°
Andar', '2860-310', 'Portugal', 'jr@acacia.pt',4),
        (5, 'Anastasia Batista', 'Coimbra', 'Praceta dos Tufos, 1513,
2° Piso', '4010-235', 'Portugal', 'ab@acacia.pt',5),
            (6, 'Deolinda Pereira Cardoso', 'Faro', 'Rua das Pombas,
77, 2° Esq', '4444-777', 'Portugal', 'dpc@acacia.pt',6),
        (7, 'Duarte Macedo da Rocha', 'Leiria', 'Edifio Capital do
Bouco, 121, 1° ', '4010-912', 'Portugal', 'dmdr@acacia.pt',7),
            (8, 'Fernando Botelho Lopes', 'Viseu', 'Urbanização das
Penedas, 61, 4°', '4919-990', 'Portugal', 'fbl@acacia.pt',8),

        (9, 'Ana Lu Bretice Renoir', 'Marseille', 'Pont de Lion, 12,
3°', '1234-567', 'France', 'albr@acacia.fr',9),
          (10, 'John Marco Holmes', 'London', 'Baker Street, 121, 1°',
'5651-127', 'UK', 'jmh@acacia.en',10)
      ;

-- Povoamento da tabela "servico"
INSERT INTO servico
      (IDServico, preco)
    VALUES
    (1,55.90)
    ;

-- Povamento da tabela "subscreve"
INSERT INTO subscreve
      (DataInicio, DataFim, preco, utilizador, servico)
    VALUES
    ('2019-06-20', '2020-06-20', 46.90, 1,1),
    ('2019-07-29', '2020-07-29', 46.90, 2,1),
    ('2019-08-14', '2020-08-14', 46.90, 3,1),
    ('2019-10-19', '2020-10-19', 46.90, 4,1),
    ('2019-12-27', '2020-12-27', 46.90, 5,1),
    ('2020-02-28', '2021-02-28', 46.90, 6,1),
    ('2020-03-20', '2021-03-20', 46.90, 7,1),
    ('2020-04-10', '2021-04-10', 55.90, 8,1),
```

```
    ('2020-04-17', '2021-04-17', 55.90, 9,1),
    ('2020-05-13', '2021-05-13', 55.90, 10,1)
    ;

-- Povamento da tabela "companhia"
INSERT INTO companhia
      (IDCompanhia, nome, comissao, dataFundacao)
    VALUES
    (1, 'BBC', 8000, '1922-10-18'),
    (2, 'HBO', 12000, '1972-11-08'),
    (3, 'SYFY', 5000, '1992-09-24'),
    (4, 'Netflix', 30000, '1997-08-27'),
    (5, 'NBC', 8000, '1926-11-15')
    ;

-- Povamento da tabela "genero"
INSERT INTO genero
      (IDGenero, genero)
    VALUES
    (1, 'drama'),
    (2,'fantasia'),
    (3, 'acao'),
    (4, 'crime'),
    (5, 'comedia')
    ;

-- Povamento da tabela "serie"
INSERT INTO serie
      (IDSerie, nome, avaliacao, classificacaoEtaria, premio, idioma,
genero, servico, companhia)
    VALUES
    (1, 'Peaky Blinders', 8.8, 18, 7, 'ingles', 4, 1,1),
    (2, 'Game of Thrones', 9.3, 18, 269, 'ingles', 3, 1,2),
    (3, 'The Magicians', 7.6, 16, 1, 'ingles', 2, 1,3),
    (4, 'La casa de Papel', 8.4, 16, 28,'espanhol', 3,1,4),
      (5, 'Lucifer', 8.2, 14, 0,'ingles', 4,1,4),
    (6, 'Baby', 6.8, 16, 0, 'italiano', 1,1,4),
    (7, 'The boys',8.7, 16, 1, 'ingles', 3,1,2),
    (8, 'Brooklyn Nine-Nine',8.4, 14, 1,'ingles', 5,1,5)
    ;
```

```
-- Povoamento da tabela "assiste"
INSERT INTO assiste
      (favorita, utilizador, serie)
    VALUES
    (true, 1,3),
    (true, 1,5),
    (false, 1,1),
    (false, 1,7),

    (true,2,4),
    (true,2,6),
    (false,2,2),
    (false,2,8),

    (true,3,1),
    (true,3,2),
    (true,3,4),
    (false,3,6),
    (false,3,7),

    (true,4,2),
    (true,4,5),
    (true,4,6),
    (false,4,1),
    (false,4,3),
    (false,4,7),
    (false,4,8),

    (true,5,3),
    (false,5,2),
    (false,5,5),

      (true,6,2),
    (true,6,4),
    (true,6,6),
    (false,6,7),
    (false,6,8),

    (false,7,1),
      (false,7,2),
    (false,7,3),
```

```
      (false,7,5),
      (false,7,7),

    (true,8,2),
    (true,8,8),
    (false,8,4),

    (true,9,1),
    (true,9,3),
    (true,9,5),
    (false,9,6),
    (false,9,8),

    (true,10,2),
    (true,10,3),
    (true,10,5),
    (true,10,7)
    ;

-- Povamento da tabela "ator"
INSERT INTO ator
      (IDAtor, nome, dataNascimento, nacionalidade, genero)
    VALUES
    (1, 'Cillian Murphy', '1976-05-25', 'irlandesa', 'masculino'),
    (2, 'Emilia Clarke', '1986-10-23', 'inglesa', 'feminino'),
    (3, 'Stella Maeve Johnston ', '1989-11-14', 'norte-americana',
'feminino'),
      (4, 'Kit Harington', '1986-12-26', 'inglesa', 'masculino'),
    (5, 'Peter Dinklage','1969-05-11', 'inglesa', 'masculino'),
    (6, 'Álvaro Morte','1975-02-23', 'espanhol', 'masculino'),
    (7, 'Úrsula Corberó','1989-08-11', 'espanhol', 'feminino'),
    (9, 'Tom Ellis',' 1978-11-17', 'ingles', 'masculino'),
    (10, 'Lauren German','1978-11-29', 'norte-americana', 'feminino'),
    (11, 'Benedetta Porcaroli','1998-06-11', 'italiana', 'feminino'),
    (12, 'Alice Pagani','1998-02-19', 'italiana', 'feminino'),
    (13, 'Karl Urban','1972-06-07', 'australiano', 'masculino'),
    (14, 'Jack Henry Quaid','1992-04-24', 'norte-americano',
'masculino'),
    (15, 'Antony Starr','1975-10-25', 'australiano', 'masculino'),
      (16, 'Andy Samberg','1978-08-18', 'norte-americano',
'masculino'),
```

```
    (17,     'Melissa     Fumero','1982-08-19',     'norte-americano',
'feminino'),
    (18,     'Andre     Braugher','1962-07-01',     'norte-americano',
'masculino'),
    (19, 'Lena Headey','1973-10-03', 'inglesa', 'feminino')
    ;

-- Povoamento da tabela possui
INSERT INTO possui
      (serie, ator)
    VALUES
    (1,1),
    (2,2),
    (2,4),
    (2,5),
    (2,19),
    (3,3),
    (4,6),
    (4,7),
    (5,9),
    (5,10),
    (6,11),
    (6,12),
    (7,13),
    (7,14),
    (7,15),
    (8,16),
    (8,17),
    (8,18)
    ;

INSERT INTO criador
      (IDCriador, nome, dataNascimento, genero, nacionalidade)
    VALUES
    (21, 'David Benioff', '1970-09-23', 'masculino', 'americano'),
    (11, 'Steven Knight', '1959-03-22', 'masculino', 'ingles'),
    (31, 'Sera Gamble', '1983-09-20', 'feminino', 'ingles'),
      (22, 'D. B. Weiss', '1971-04-23', 'masculino', 'americano'),
      (41, 'Álex Pina', '1967-06-23', 'masculino', 'espanhol'),
      (51, 'Tom Kapinos', '1969-07-12', 'masculino', 'americano'),
      (61, 'Antonio Le Fosse', '1952-03-22', 'masculino', 'italiano'),
```

```
      (71, 'Eric Kripke', '1974-04-24', 'masculino', 'americano'),
      (81, 'Dan Goor', '1975-04-28', 'masculino', 'americano')
    ;

-- Povoamento da tabela "cria"
INSERT INTO cria
      (serie,criador)
    VALUES
    (1,11),
    (2,21),
    (2,22),
    (3,31),
      (4,41),
      (5,51),
      (6,61),
      (7,71),
      (8,81)
    ;

-- Povoamento da tabela legenda
INSERT INTO legendas
      (IDLegenda, legendas)
    VALUES
    (1,'ingles'),
    (2,'portugues'),
    (3,'espanhol')
    ;


-- Povamento da tabela "temporada"
INSERT INTO temporada
      (IDTemporada, numero, ano, serie)
    VALUES
    (11,1,2013,1),
    (12,2,2014,1),
    (21,1,2011,2),
    (22,2,2012,2),
    (31,1,2015,3),
    (41,1,2017,4),
    (51,1,2019,5),
    (61,1,2018,6),
```

```
    (62,2,2019,6),
    (63,3,2020,6),
    (71,1,2019,7),
    (72,2,2020,7),
    (87,7,2020,8)
    ;


-- Povamento da tabela "episodio"
INSERT INTO episodio
      (IDEpisodio, nome, avaliacao, numero, duracao, legendas,
temporada)
    VALUES
      (111,'Episódio #1.1', 8.2, 1, 50, 1,11),
      (112,'Episódio #1.2', 8.4, 2, 50, 1,11),
      (113,'Episódio #1.3', 8.3, 3, 50, 1,11),
      (114,'Episódio #1.4', 8.7, 4, 50, 1,11),
      (115,'Episódio #1.5', 9.0, 5, 50, 1,11),
      (116,'Episódio #1.6', 9.2, 6, 50, 1,11),
      (121,'Episódio #2.1', 8.6, 1, 50, 1,11),
      (122,'Episódio #2.2', 8.5, 2, 50, 1,11),
      (123,'Episódio #2.3', 8.7, 3, 50, 1,11),
      (124,'Episódio #2.4', 8.6, 4, 50, 1,11),
      (125,'Episódio #2.5', 8.8, 5, 50, 1,11),
      (126,'Episódio #2.6', 9.6, 6, 50, 1,11),
    (211,'Winter Is Coming', 9.1, 1, 55, 2,21),
    (212,'The Kingsroad', 8.8, 2, 55, 2,21),
    (213,'Lord Snow', 8.7, 3, 55, 2,21),
    (214,'Cripples, Bastards, and Broken Things', 8.8, 4, 55, 2,21),
    (215,'The Wolf and the Lion', 9.1, 5, 55, 2,21),
    (216,'A Golden Crown', 9.2, 6, 55, 2,21),
    (217,'You Win or You Die', 9.2, 7, 55, 2,21),
    (218,'The Pointy End', 9.0, 8, 55, 2,21),
    (219,'Baelor', 9.6, 9, 55, 2,21),
    (2110,'Fire and Blood', 9.5, 10, 55, 2,21),
    (221,'The North Remembers', 8.8, 1, 55, 2,22),
    (222,'The Night Lands', 8.5, 2, 55, 2,22),
    (223,'What Is Dead May Never Die', 8.8, 3, 55, 2,22),
    (224,'Garden of Bones', 8.8, 4, 55, 2,22),
    (225,'The Ghost of Harrenhal', 8.8, 5, 55, 2,22),
    (226,'The Old Gods and the New', 9.1, 6, 55, 2,22),
```

```
(227,'A Man Without Honor', 8.9, 7, 55, 2,22),
(228,'The Prince of Winterfell', 8.8, 8, 55, 2,22),
(229,'Blackwater', 9.7, 9, 55, 2,22),
(2210,'Valar Morghulis', 9.4, 10, 55, 2,22),
(311,'Unauthorized Magic', 8.0, 1, 48, 2,31),
(312,'The Source of Magic', 7.6, 2, 48, 2,31),
(313,'Consequences of Advanced Spellcasting', 7.7, 3, 48, 2,31),
(314,'The World in the Walls', 8.3, 4, 48, 2,31),
(315,'Mendings, Major and Minor', 7.9, 5, 48, 2,31),
(316,'Impractical Applications', 8.2, 6, 48, 2,31),
(317,'The Mayakovsky Circumstance', 8.1, 7, 48, 2,31),
(318,'The Strangled Heart', 8.1, 8, 48, 2,31),
(319,'The Writing Room', 8.6, 9, 48, 2,31),
(3110,'Homecoming', 8.1, 10, 48, 2,31),
(3111,'Remedial Battle Magic', 8.0, 11, 48, 2,31),
(3112,'Thirty-Nine Graves', 8.2, 12, 48, 2,31),
(3113,'Have You Brought Me Little Cakes', 8.6, 13, 48, 2,31),
(411,'Efectuar lo acordado', 8.3, 1, 45, 2,41),
(412,'Imprudencias letales', 8.4, 2, 45, 2,41),
(413,'Errar al disparar', 8.2, 3, 45, 2,41),
(414,'Caballo de Troya', 8.3, 4, 45, 2,41),
(415,'El día de la marmota', 8.4, 5, 45, 2,41),
(416,'La cálida Guerra Fría', 8.3, 6, 45, 2,41),
(417,'Refrigerada inestabilidad', 8.4, 7, 45, 2,41),
(418,'Tú lo has buscado', 8.2, 8, 45, 2,41),
(419,'El que la sigue la consigue', 8.7, 9, 45, 2,41),
(511,'Pilot', 8.8, 1, 40, 2,51),
(512,'Lucifer, Stay. Good Devil.', 8.2, 2, 40, 2,51),
(513,'The Would-Be Prince of Darkness', 8.2, 3, 40, 2,51),
(514,'Manly Whatnots', 8.6, 4, 40, 2,51),
(515,'Sweet Kicks', 8.1, 5, 40, 2,51),
(516,'Favorite Son', 8.8, 6, 40, 2,51),
(517,'Wingman', 8.7, 7, 40, 2,51),
(518,'Et Tu, Doctor?', 8.3, 8, 40, 2,51),
(519,'A Priest Walks Into a Bar', 9.1, 9, 40, 2,51),
(5110,'Pops', 8.4, 10, 40, 2,51),
(5111,'St. Lucifer', 8.8, 11, 40, 2,51),
(5112,'#TeamLucifer', 9.1, 12, 40, 2,51),
(5113,'Take Me Back to Hell', 9.2, 13, 40, 2,51),
(611,'Superpoteri', 6.9, 1, 30, 2,61),
(612,'Burattino', 7.1, 2, 30, 2,61),
```

```
    (613,'#Friendzone', 7.2, 3, 30, 2,61),
    (614,'Emma', 7.2, 4, 30, 2,61),
    (615,'L ultimo scatto', 7.2, 5, 30, 2,61),
    (616,'#Love', 7.3, 6, 30, 2,61),
    (621,'#justagame', 7.6, 1, 30, 2,62),
    (622,'Rilancio', 7.9, 2, 30, 2,62),
    (623,'Fantasmi', 7.9, 3, 30, 2,62),
    (625,'Vicolo cieco', 7.9, 5, 30, 2,62),
    (626,'Baby', 8.1, 6, 30, 2,62),
    (631,'San Valentino', 7.6, 1, 30, 2,63),
    (632,'26 aprile 1915', 7.7, 2, 30, 2,63),
    (633,'Esprimi un desiderio', 8.0, 3, 30, 2,63),
    (634,'Niente più segreti', 7.6, 4, 30, 2,63),
    (635,'100 giorni', 7.7, 5, 30, 2,63),
    (636,'Oltre l acquario', 8.0, 6, 30, 2,63),
    (711,'The Name of the Game', 8.8, 1, 54, 2,71),
    (712,'Cherry', 8.6, 2, 54, 2,71),
    (713,'Get Some', 8.5, 3, 54, 2,71),
    (714,'The Female of the Species', 8.8, 4, 54, 2,71),
    (715,'Good for the Soul', 8.5, 5, 54, 2,71),
    (716,'The Innocents', 8.3, 6, 54, 2,71),
    (717,'The Self-Preservation Society', 8.9, 7, 54, 2,71),
    (718,'You Found Me', 9.1, 8, 54, 2,71),
    (721,'The Big Ride', 8.3, 1, 54, 2,72),
    (722,'Proper Preparation and Planning', 7.9, 2, 54, 2,72),
    (723,'Over the Hill with the Swords of a Thousand Men', 9.1, 3,
54, 2,72),
    (724,'Nothing Like It in the World', 8.1, 4, 54, 2,72),
    (725,'We Gotta Go Now', 8.5, 5, 54, 2,72),
    (726,'The Bloody Doors Off', 9.1, 6, 54, 2,72),
    (727,'Butcher, Baker, Candlestick Maker', 9.1, 7, 54, 2,72),
    (728,'What I Know', 9.5, 7, 54, 2,72),
    (871,'Manhunter', 8.2, 1, 32, 2,87),
    (872,'Captain Kim', 8.4, 2, 32, 2,87),
    (873,'Pimemento', 8.6, 3, 32, 2,87),
    (874,'The Jimmy Jab Games II', 8.0, 4, 32, 2,87),
    (875,'Debbie', 7.4, 5, 32, 2,87),
    (876,'Trying', 8.0, 6, 32, 2,87),
    (877,'Ding Dong', 8.8, 7, 32, 2,87),
    (878,'The Takeback', 8.3, 8, 32, 2,87),
    (879,'Dillman', 8.8, 9, 32, 2,87),
```

```
(8710,'Admiral Peralta', 8.0, 10, 32, 2,87),
(8711,'Valloweaster', 8.4, 11, 32, 2,87),
(8712,'Ransom', 8.9, 12, 32, 2,87),
(8713,'Lights Out', 9.3, 13, 32, 2,87)
;
```

III. Anexo 3 – *Script* de interrogações

```
-- Query 1) Identificar qual o top N das série mais vista.
DELIMITER $$
CREATE PROCEDURE topNSeriesMaisVistas (IN topN INT)
BEGIN
SELECT S.nome, count(A.utilizador) AS nrUtilizadores
      FROM assiste AS A
            JOIN serie AS S ON S.IDSerie = A.serie
            GROUP BY A.serie
                  ORDER BY nrUtilizadores DESC
                  LIMIT topN;
END $$

-- Query 2) Identificar qual o top N de series com melhor classificação.
DELIMITER $$
CREATE PROCEDURE topNSerieMelhorClassificada(IN topN INT)
BEGIN
SELECT nome, avaliacao
      FROM serie
            ORDER BY avaliacao DESC
                  LIMIT topN;
END $$

-- Query 3) O sistema deverá ser capaz de identificar quais as companhias que forneceram mais séries.
DELIMITER $$
CREATE PROCEDURE MaioresCompanhias()
BEGIN
SELECT C.nome, count(S.companhia) AS nrSeries
      FROM serie AS S
            JOIN companhia AS C ON C.IDCompanhia = S.companhia
                  GROUP BY S.companhia
                  ORDER BY nrSeries DESC ;
```

```
END $$


-- Query 4) O sistema deverá ser capaz de apresentar todas as series e
o seu numero de premios.
DELIMITER $$
CREATE PROCEDURE TopSeriesPremios()
BEGIN
SELECT S.nome, S.premio
      FROM serie AS S
            ORDER BY S.premio DESC;
END $$


-- Query 5) O sistema deverá ser capaz de identificar qual é top N dos
utilizadores que vê mais séries.
DELIMITER $$
CREATE PROCEDURE NrSeriesPorUtilizador(IN topN INT)
BEGIN
SELECT U.nome , count(A.serie) AS nrSeries
 FROM assiste AS A
      JOIN utilizador AS U ON U.IDUtilizador = A.utilizador
            GROUP BY A.utilizador
                  ORDER BY nrSeries DESC
                        LIMIT topN;
END $$


-- Query 6) O sistema deverá ser capaz de identificar as séries de um
determinado idioma.
DELIMITER $$
CREATE PROCEDURE ProcuraSeriesNacionalidade(IN idioma VARCHAR(65))
BEGIN
SELECT S.idioma, S.nome
      FROM serie AS S
            WHERE S.idioma = idioma; -- exemplo: ingles, italiano,
espanhol
END $$
```

```
-- Query 7) O sistema deverá ser capaz de identificar séries cujos episódios possuem legendas de um determinado idioma.
DELIMITER $$
CREATE PROCEDURE ProcuraSeriesTraduzidas(IN legenda VARCHAR(65))
BEGIN
SELECT S.nome, COUNT(serie) AS Episodios_Legendados, L.legendas
      FROM serie AS S
            JOIN temporada AS T ON T.serie = S.IDSerie
        JOIN episodio AS E ON T.IDTemporada = E.temporada
        JOIN legendas AS L ON L.IDLegenda = E.legendas
                  WHERE L.legendas = legenda   -- exemplo: ingles, portugues, espanhol
                        GROUP BY S.nome;
END $$


-- Query 8) O sistema deverá ser capaz de identificar quais as séries apropriadas para uma determinada faixa etária.
DELIMITER $$
CREATE PROCEDURE ConsultaSeriesIdade(IN idade INT)
BEGIN
SELECT S.nome , S.classificacaoEtaria
      FROM serie AS S
            WHERE S.classificacaoEtaria <= idade; -- exemplo: 12, 14, 16, 18
END $$


-- Query 9) O sistema deverá ser capaz de identificar quais as séries favoritas de um utilizador.
DELIMITER $$
CREATE PROCEDURE SeriesFavoritasUtilizador(IN idUtilizador INT)
BEGIN
SELECT U.nome, S.nome AS SeriesFavoritas
      FROM assiste AS A
                  JOIN serie AS S ON A.serie = S.IDSerie
            JOIN utilizador AS U ON A.utilizador = U.IDUtilizador
            WHERE A.favorita = 1
                              && A.utilizador = idUtilizador;  -- exemplo: 1,..,10
END $$
```

```
-- Query 10) O sistema deverá ser capaz de identificar qual o numero de utilizadores com conta ativa.
DELIMITER $$
CREATE PROCEDURE UtilizadoresAtivos()
BEGIN
SELECT U.nome, S.DataInicio, S.DataFim
      FROM subscreve AS S
    JOIN utilizador AS U ON U.IDUtilizador = S.utilizador
            WHERE S.DataFim >= current_date();
END $$


-- Query 11) O sistema deverá ser capaz de identificar qual a faturação total.
DELIMITER $$
CREATE PROCEDURE FaturacaoTotal()
BEGIN
SELECT SUM(S.preco) AS Faturação
      FROM subscreve AS S
            GROUP BY S.servico;
END $$
```

IV. Anexo 4 – *Script* de criação de Vistas

```
-- Vista query 1
CREATE VIEW vwtop10SeriesMaisVistas AS
SELECT S.nome, count(A.utilizador) AS nrUtilizadores
      FROM assiste AS A
            JOIN serie AS S ON S.IDSerie = A.serie
            GROUP BY A.serie
                  ORDER BY nrUtilizadores DESC
                  LIMIT 10;

-- Vista query 2
CREATE VIEW vwtop6MelhorClassificada AS
SELECT nome, avaliacao
      FROM serie
            ORDER BY avaliacao DESC
                  LIMIT 6;

-- Vista query 3
CREATE VIEW vwmaiorescompanhias AS
SELECT C.nome, count(S.companhia) AS nrSeries
      FROM serie AS S
            JOIN companhia AS C ON C.IDCompanhia = S.companhia
                   GROUP BY S.companhia
                   ORDER BY nrSeries DESC ;

-- Vista query 4
CREATE VIEW vwtopSeriesPremios AS
SELECT S.nome, S.premio
      FROM serie AS S
            ORDER BY S.premio DESC;


-- Vista query 5
```

```
CREATE VIEW vwtop5SeriesUtilizador AS
SELECT U.nome , count(A.serie) AS nrSeries
 FROM assiste AS A
      JOIN utilizador AS U ON U.IDUtilizador = A.utilizador
            GROUP BY A.utilizador
                  ORDER BY nrSeries DESC
                        LIMIT 5;


-- Vista query 6
CREATE VIEW vwSeriesIdioma AS
SELECT S.idioma, S.nome
      FROM serie AS S
            WHERE S.idioma = 'ingles'; -- exemplo: ingles, italiano,
espanhol


-- Vista query 7
CREATE VIEW vwseriestraduzidas AS
SELECT S.nome, COUNT(serie) AS Episodios_Legendados, L.legendas
      FROM serie AS S
            JOIN temporada AS T ON T.serie = S.IDSerie
        JOIN episodio AS E ON T.IDTemporada = E.temporada
        JOIN legendas AS L ON L.IDLegenda = E.legendas
                  WHERE L.legendas = 'portugues'   -- exemplo: ingles,
portugues, espanhol
                        GROUP BY S.nome;


-- Vista query 8
CREATE VIEW vwclassificacaoEtaria AS
SELECT S.nome , S.classificacaoEtaria
      FROM serie AS S
            WHERE S.classificacaoEtaria <= 16;


-- Vista query 9
CREATE VIEW vwSeriesFavoritas AS
SELECT U.nome, S.nome AS SeriesFavoritas
      FROM assiste AS A
                  JOIN serie AS S ON A.serie = S.IDSerie
```

```
            JOIN utilizador AS U ON A.utilizador = U.IDUtilizador
            WHERE A.favorita = 1
                              && A.utilizador = 1; -- exemplo: 1,..,10


-- Vista query 10
CREATE VIEW vwUtilizadoresAtivos AS
SELECT U.nome, S.DataInicio, S.DataFim
      FROM subscreve AS S
    JOIN utilizador AS U ON U.IDUtilizador = S.utilizador
            WHERE S.DataFim >= current_date();


-- Vista query 11
CREATE VIEW vwFaturacaoTotal AS
SELECT SUM(S.preco) AS Faturação
      FROM subscreve AS S
            GROUP BY S.servico;
```