

**2016/2017 – 2º Semestre**

**MIEIC**

**Trabalho realizado por:**

Grupo 101

|  |  |
| --- | --- |
| Afonso Jorge Ramos  Diogo Filipe Dores  José Pedro Borges | [up201506239](mailto:up201506239@fe.up.pt)  [up201504614](mailto:up201504614@fe.up.pt)  [up201503603](mailto:up201503603@fe.up.pt) |

**Bases de Dados**

TVShow Time – Parte II

**Índice**

[Breve Descrição do Projeto 2](#_Toc483778789)

[Diagrama UML Parte 1 3](#_Toc483778790)

[Diagrama UML Parte 2 4](#_Toc483778791)

[Esquema Relacional 5](#_Toc483778792)

[Restrições 8](#_Toc483778793)

[Show 8](#_Toc483778794)

[Season 8](#_Toc483778795)

[Episode 9](#_Toc483778796)

[Actor 9](#_Toc483778797)

[Character 9](#_Toc483778798)

[Message 10](#_Toc483778799)

[Comment 10](#_Toc483778800)

[Watched 11](#_Toc483778801)

[Country 11](#_Toc483778802)

[PremiumUser 11](#_Toc483778803)

[User 12](#_Toc483778804)

[Gatilhos 13](#_Toc483778805)

[Interrogações 13](#_Toc483778806)

Breve Descrição do Projeto

Para a realização deste projeto decidimos basearmo-nos na base de dados de uma aplicação de “gestão” de séries televisivas – *TVShowTime*. O objetivo desta plataforma é, de uma forma geral, construir um sistema de informação que guarde na sua plataforma todos os episódios já vistos e por ver por parte dos amantes de séries televisivas. No entanto, existem certas estatísticas que não sabemos como são calculadas, ou, de onde vêm, como por exemplo o rating por episódio existente na plataforma e, por isso, decidimos cingir-nos ao desenvolvimento de todas as funcionalidades base.

Neste sistema encontramos vários **utilizadores** que têm a capacidade de seguir uma **série**. Estas podem ser subdivididas em **temporadas**, que, por sua vez, são compostas por **episódios.** Além disso, a cada série estão associados vários **atores**, os quais estão associados a uma certa série pelo nome da personagem que interpretam.

Entre dois utilizadores é possível existirem dois tipos de relações. Estes podem seguir-se mutuamente, podendo assim saber quais os últimos episódios que o outro utilizador visualizou ou então podem simplesmente falar entre si através de mensagens pessoais.

Existem dois tipos de utilizadores. Os **gratuitos**, que têm acesso às funcionalidades básicas e os ***premium***, que todos os meses pagam uma certa quantia, mas podem aceder a todas as funcionalidades da aplicação.

Os utilizadores e as séries estão relacionados de duas formas: cada utilizador pode seguir inúmeras séries, no entanto, apenas pode definir entre 0 a 6 séries como as suas favoritas.

Concluindo, de modo a que um utilizador possa gerir as séries que está, de momento, a seguir, existe uma relação entre o utilizador e os episódios, que regista se este já visualizou certo episódio ou não e se esse mesmo episódio já foi lançado ou não. Um utilizador tem ainda a possibilidade de deixar a sua opinião e previsões para que outros utilizadores possam responder a esta da mesma forma relativamente a um episódio em específico.

Diagrama UML Parte 1

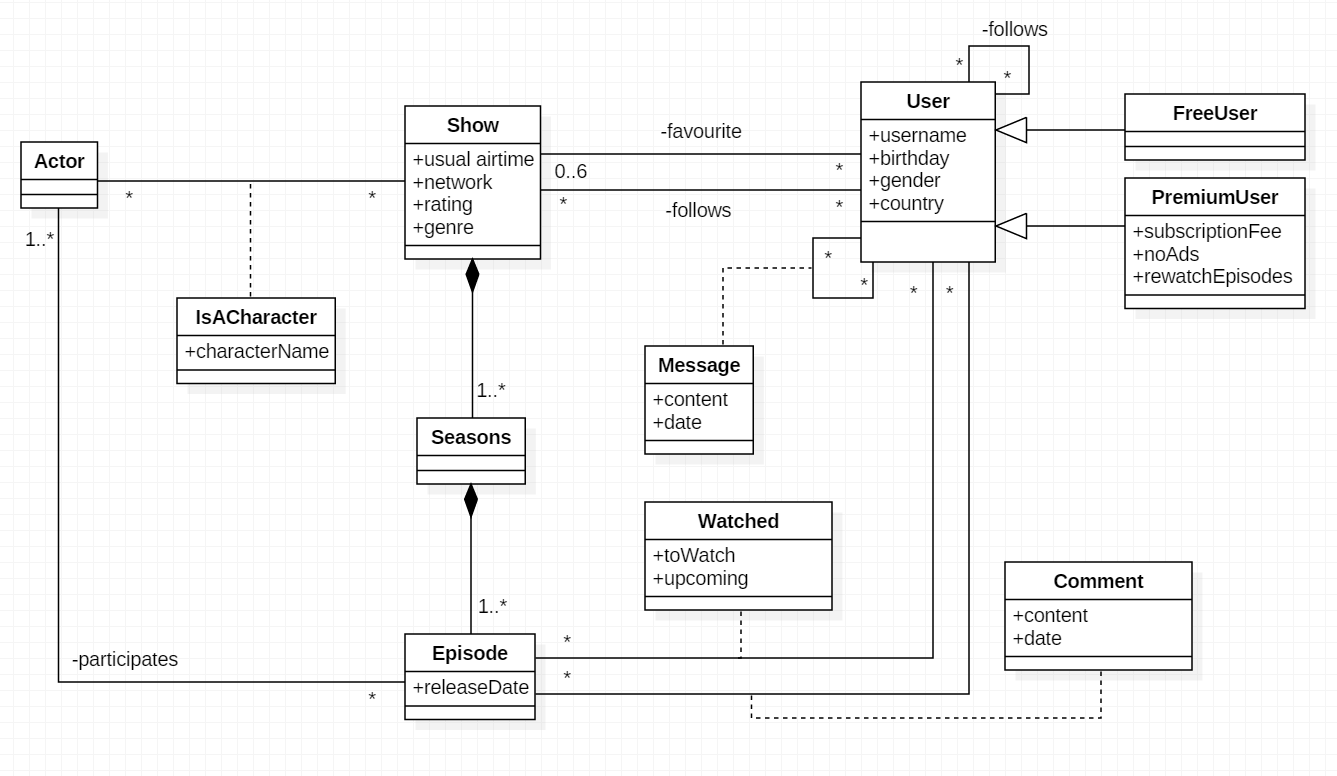


Figura 1 - Diagrama UML

Diagrama UML Parte 2

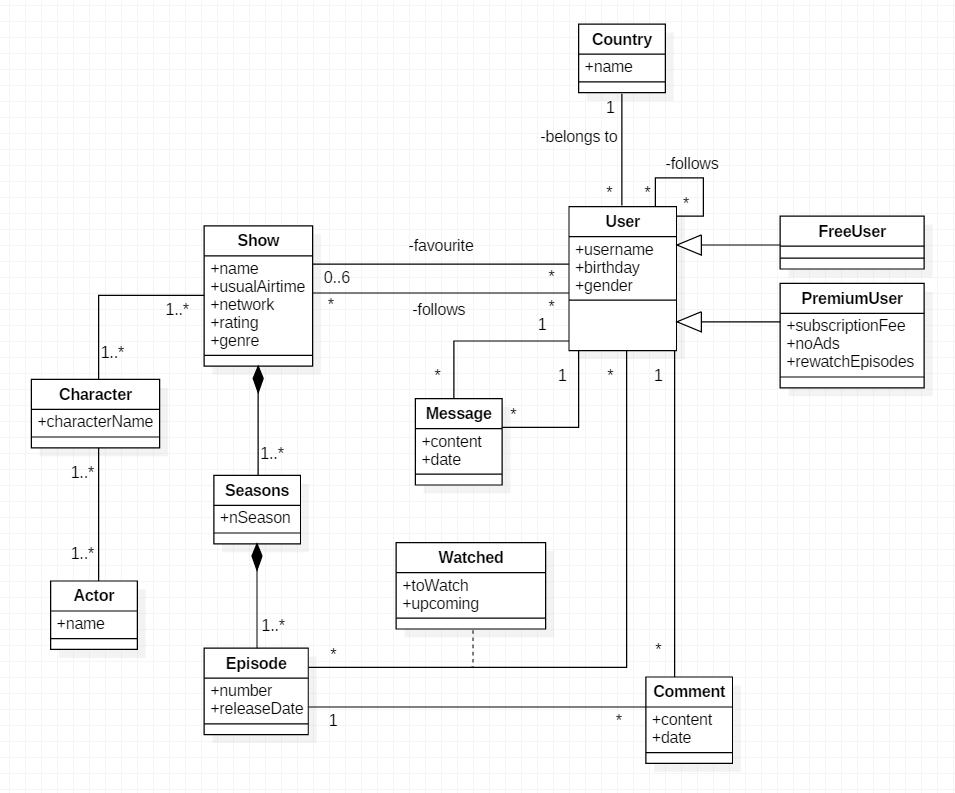


Figura 2 – UML atualizado após revisões propostas pelo professor.

Esquema Relacional

**User (idUser, username, birthday, gender, idCountry->Country, userType)**

{idUser} -> {username, birthday, gender, idCountry, userType)

**FavoriteShow (idUser->User, idShow->Show)**

**FollowedShow (idUser->User, idShow->Show)**

**PremiumUser (idUser->User,subscriptionFee, noAds, rewatchEpisodes)**

{idUser} -> {subscriptionFee, noAds, rewatchEpisodes}

**Show (idShow, name, usualAirtime, network, rating, genre)**

{idShow} -> {name, usualAirtime, network, rating, genre}

**Season (idSeason, nSeason, idShow->Show)**

{idSeason}-> {nSeason, idShow}

**Episode (idEpisode, epNumber, releaseDate, idSeason->Season)**

{idEpisode} -> {epNumber, releaseDate, idSeason}

**FollowedUser (idUser1->User1, idUser2->User2)**

**Actor (idActor, name)**

{idActor} -> {name}

**ActorAsCharacter (idActor->Actor, idCharacter->Character)**

**Character (idCharacter, characterName)**

{idCharacter} -> {characterName}

**CharacterInShow (idCharacter->Character, idShow->Show)**

**Message(idMessage,content,msgDate,idUser1->User,idUser2->User)**

{idMessage} -> {content, msgDate, idUser1, idUser2}

**Comment(idComment,content,cmtDate,idUser->User,idEpisode->Episode)**

{idComment} -> {content, cmtDate, idUser, idEpisode}

**Watched (toWatch, upcoming, idEpisode->Episode, idUser->User)**

{idEpisode, idUser} -> {toWatch, upcoming}

**Country (idCountry, name)**

{idCountry} -> {name}

**Forma Normal de Boyce-Codd**

*Para cada dependência funcional* Ᾱ →𝐵̅ *não trivial na relação, Ᾱ é superchave.*

**3ª Forma Normal**

*A relação está na BCNF ou, para cada dependência funcional* Ᾱ →𝐵̅ *não trivial,* 𝐵̅ *consiste apenas em atributos primos.*

Daqui podemos tirar que, para assegurar que uma relação está na 3ª Forma Normal, e necessário certificarmos-mos que todos os atributos presentes na secção direita das dependências funcionais existentes são primos, sendo também possível que em todas as dependências funcionais, os atributos existentes na porção esquerda serem superchaves.

Se quisermos garantir que uma relação esteja na Formal Normal de Boyce-Codd, é necessário certificar que para além de se encontrar na 3ª Forma Normal, desta forma assegurando que todas as dependências existentes têm à sua esquerda uma superchave.

Desta forma, qualquer relação existente no nosso esquema está na 3ª Forma Normal e na Forma Normal de Boyce-Codd pois, em qualquer dependência funcional, à esquerda surge a chave, e todos os atributos existentes nas relações são primos.

Restrições

Para melhor representar a forma de implementação de restrições, decidimos demonstrá-las em tabelas. Cada tabela representa uma classe. Por sua vez, as colunas representam (da esquerda para a direita) as variáveis da classe a que pertencem, uma breve descrição do porquê de uma certa restrição ter sido cridas e como foi implementada em SQL.

|  |  |  |
| --- | --- | --- |
| Show | | |
| name | Se uma série existe, então terá que ter um nome. | name TEXT NOT NULL |
| usualAirtime | A hora a que a série é lançada. Esta nunca poderá ser nula. | usualAirtime TIME NOT NULL |
| network | Se um canal existe, então terá que ter um nome. | network TEXT NOT NULL |
| rating | A classificação da série. Esta varia de 0.0 a 5.0. | rating REAL CHECK  (rating >=0.0 and rating <=5.0) |
| genre | Uma string que guarda o tipo de conteúdo da série. Esta nunca poderá ser vazia. | genre TEXT NOT NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Season | | | | |
| nSeason | O número de temporadas. Este nunca poderá ser nulo. | | nSeason INTEGER NOT NULL | |
| idShow | Uma *foreign key* associada à classe *Show*. É utilizada uma vez que uma série está subdividida em várias temporadas. | | idShow INTEGER NOT NULL REFERENCES Show(idShow) | |
| Episode | | | | |
| epNumber | | O número do episódio. Este nunca poderá ser nulo. | | epNumber INTEGER NOT NULL |
| releaseDate | | A data de lançamento do episódio. Esta nunca poderá ser nula. | | releaseDate DATE NOT NULL |
| idSeason | | Uma *foreign key* associada à classe *Season*. É utilizada uma vez que uma temporada é composta por vários episódios. | | idSeason INTEGER NOT NULL REFERENCES Season(idSeason) |

|  |  |  |
| --- | --- | --- |
| Actor | | |
| name | Se um ator existe, então terá que ter um nome. | name TEXT NOT NULL |

|  |  |  |
| --- | --- | --- |
| Character | | |
| characterName | Se uma personagem existe, então terá que ter um nome. | characterName TEXT NOT NULL |
| idActor | Uma *foreign key* associada à classe *Actor*. É utilizada uma vez que uma personagem é representada por um certo ator. | idActor INTEGER NOT NULL REFERENCES Actor(idActor) |
| idShow | Uma *foreign key* associada à classe *Show*. É utilizada uma vez que uma personagem pertence a uma determinada série. | idShow INTEGER NOT NULL REFERENCES Show(idShow) |

|  |  |  |
| --- | --- | --- |
| Message | | |
| content | Uma mensagem nunca poderá ter um conteúdo vazio. | content TEXT NOT NULL |
| msgDate | A data a que uma mensagem foi enviada. Esta nunca poderá ser nula. | msgDate DATE NOT NULL |
| idUser1/  idUser2 | Uma *foreign key* associada à classe *User*. É utilizada uma vez que uma mensagem é trocada entre dois utilizadores. | idUser1/2 INTEGER NOT NULL REFERENCES User(idUser) |

|  |  |  |
| --- | --- | --- |
| Comment | | |
| content | Um comentário nunca poderá ter um conteúdo vazio. | content TEXT NOT NULL |
| msgDate | A data a que um comentário foi enviado. Esta nunca poderá ser nula. | cmtDate DATE NOT NULL |
| idUser | Uma *foreign key* associada à classe *User*. É utilizada uma vez que um comentário é feito por um utilizador. | idUser INTEGER NOT NULL REFERENCES User(idUser) |
| idEpisode | Uma *foreign key* associada à classe *Episode*. É utilizada uma vez que um comentário é feito num certo episódio. | idEpisode INTEGER NOT NULL REFERENCES Episode(idEpisode) |

|  |  |  |
| --- | --- | --- |
| Watched | | |
| toWatch | Se um episódio já tiver sido visto terá um valor de 0. Caso contrário será 1. | toWatch INTEGER CHECK  (toWatch=1 OR toWatch=0) |
| upcoming | Se um episódio já tiver sido visto terá um valor de 1. Caso contrário será 0. | upcoming INTEGER CHECK  (upcoming=1 OR upcoming=0) |
| idEpisode | Uma *foreign key* associada à classe *Episode*. É utilizada uma vez que o episódio é que será marcado como visto ou não. | idEpisode INTEGER NOT NULL REFERENCES Episode(idEpisode) |
| idUser | Uma *foreign key* associada à classe *User*. É utilizada uma vez que um utilizador marca um certo episódio como visto. | idUser INTEGER NOT NULL REFERENCES User(idUser) |

|  |  |  |
| --- | --- | --- |
| Country | | |
| name | O país de um utilizador. Nunca poderá ser nulo | name TEXT NOT NULL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PremiumUser | | | | |
| subscriptionFee | | Como nos estamos a referir a um utilizador premium, o valor desta variável terá que ser 1. | | subscriptionFee INTEGER DEFAULT 1 |
| noAds | | Como nos estamos a referir a um utilizador premium, o valor desta variável terá que ser 1. | | noAds INTEGER DEFAULT 1 |
| rewatchEpisodes | | Como nos estamos a referir a um utilizador premium, o valor desta variável terá que ser 1. | | rewatchEpisodes INTEGER DEFAULT 1 |
| User | | | | |
| username | Um username único representativo de um utilizador. Se este existe, então não poderá ser nulo. | | username TEXT UNIQUE NOT NULL | |
| birthday | A data de nascimento de um utilizador. Esta nunca poderá ser nula. | | birthday DATE NOT NULL | |
| gender | Uma variável de 1 até 6 caracteres. Apenas pode ter os valores “Male” ou “Female”. | | gender varchar(6) CHECK  (gender=”Male” or gender=”Female”) | |
| userType | Se um utilizador for do tipo premium a variável terá um valor de 1. Caso contrário, terá um valor de 0. | | userType INTEGER CHECK  (userType=1 OR userType=0) | |
| idCountry | Uma *foreign key* associada à classe *Country*. É utilizada uma vez que um utilizador indica o seu país na altura de registo na *app*. | | iCountry INTEGER NOT NULL REFERENCES Country(idCountry) | |
| Classes derivadas | *User* tem duas classes derivadas: *FreeUser* e *PremiumUser*. Ambas possuem uma *foreign key* que está associada a esta classe. | | idUser INTEGER REFERENCES User(idUser) | |

Gatilhos

1. Atribuição de rating base para um novo Show;
2. Impede o utilizador de marcar um episódio que ainda não foi lançado como visto, retornando erro;
3. Filtro de linguagem para comentários retirando a palavra não apropriadas.

Interrogações

1. Listar series por rating;
2. Séries de animação que dão antes das 8 da manhã;
3. Comentários deixados por *users premium*;
4. Comentários deixados no mesmo dia em que o episódio foi lançado;
5. Países ordenados por numero de utilizadores;
6. Utilizadores ordenador por mensagens enviadas no verão de 2012;
7. Número de personagens interpretadas por cada ator;
8. Networks com maior rating;
9. Idade média dos utilizadores que seguem *Game Of Thrones*;
10. Séries recomendadas a quem vê séries de comédia ordenadas por rating;
11. Séries vistas pelo utilizador com id 6.