Aitor Juncal Blanco

ANOTACIONES JPA

1-) Usuario

La clase usuario, la he anotado como Entity, utilizando el nombre del usuario como id, ya que este es único a lo largo del sistema. Por otro lado, respecto a las asociaciones con otras clases tenemos estas 5:

- seriesPendientes / seriesEmpezadas / seriesTerminadas : En este caso, se modela una relación ManyToMany con la clase Serie, ya que un usuario puede ver múltiples series, y una serie puede ser vista por múltiples usuarios.
- capitulosVistos : En este caso, la relacion también es ManyToMany por la misma razón que antes, un capítulo puede haber sido visto por múltiples usuarios, así como un usuario puede ver múltiples capítulos. En este caso, como la clase CapitulosVistos tiene una propiedad usuario, será necesario hacer un mappedBy del mismo
- Facturas : Aquí la relación es OneToMany, ya que por un lado un usuario puede tener varías facturas, tantas como meses haya hecho algún consumo, mientras que para cada factura sabemos cuál es el usuario a la que va dirigida. Además, para esta estructura he utilizado la etiqueta (cascade = CascadeType.ALL), para reflejar que un cambio en el usuario también puede implicar cambios en las facturas, por ej cambiar la suscripción. Por otro lado, también he puesto la etiqueta (mappedBy = “usuario”), ya que como dije antes, cada factura tiene que saber al usuario a la que va dirigida

Por último, es importante mencionar que en todas estas estructuras, así como en el resto de asociaciones OneToMany o ManyToMany que se puedan usar en otras clases, la estrategia de fetch.type es siempre lazy por defecto.

2-) Serie

Esta clase también es Entity, y como tal, tiene un identificador “id” generado mediante la estrategia Identity, por lo que es generado por el gestor de base de datos incrementalmente. Por otro lado, las asociaciones con otras clases son las siguientes:

- temporadas : lista de temporadas pertenecientes a una serie, se trata de una asociación oneToMany ya que una serie puede tener varias temporadas, pero una temporada solo puede pertenecer a una serie.Además, se utiliza mappedBy=”temporada” ya que un capítulo sabe a qué temporada pertenece. Por último, también se aplica la estrategia cascade para que se reflejen los cambios en la relación series->temporadas.
- listaActores / listaAutores : Simples listas de strings, por lo que la única etiqueta válida para este caso es @ElementCollection
- categoria : este es un objeto de la clase Categoria, que es un simple enum, por lo que utilizo la etiqueta @Embedded para cargarlo.

3-) Temporada

Esta clase, al igual que las anteriores, es Entity, y tiene un id también generado mediante la estrategia identity. Por otro lado, las asociaciones con otras clases son estas dos:

- capitulos : relación OneToMany, ya que una temporada puede tener varios capítulos, pero un capítulo pertenece unívocamente a una temporada. Por otro lado, aquí también se utiliza la operacion cascadeType.ALL, así como mappedBy=”temporada”.
- serie : relación ManyToOne(), ya que una temporada solo puede pertenecer a una serie, esta relación es la inversa a “temporadas” en la clase serie, y este es el elemento que mapeabamos ahí.

4-) Capitulo

Esta clase también es Entity, y su id es generado mediante la estrategia identity. Por último, la única asociación con otras clases es la variable “temporada”, que refleja a que temporada a la que pertenece el capítulo, y se trata de una relación ManyToOne(), ya que es la relación inversa a la lista de capítulos en la clase temporada.

4-) CapitulosVistos

Esta clase también es Entity, y su id es generado mediante la estrategia identity. Además, la única asociación con otras clases es el usuario al que pertenece esta lista de capítulos vistos, y se trata de una relación ManyToOne(). Mientras que en este caso la única relación existente es con la clase capítulos, es una relación OneToMany() que muestra los capítulos que han sido vistos por un usuario.

5-) Categoria

Esta clase es un simple enum que puede ser cargado por la clase serie, por lo que, para ser coherente con el hecho de que en la clase serie se carga la categoria como @Embedded, en este caso la clase categoria deberá ser @Embeddable.

6-) Factura

Esta clase vuelve a ser Entity, y su id también es generado mediante la estrategia identity. Mientras que sus únicas asociaciones con otras clases son estados dos :

- cargos : lista de cargos perteneciente a una factura. Si no tenemos el plan premium, cada capítulo visto es un cargo. Además , la relación es OneToMany() ya que una factura puede tener varios cargos, pero un cargo solo puede pertenecer a una factura. Por último, también se ha aplicado la estrategia cascadeType.All para reflejar los cambios entre la factura y sus cargos.
- usuario : relación ManyToOne, ya que es la relación inversa de “facturas” en la clase Usuario.

7-) Cargo

Esta clase, al igual que la mayoría de las anteriores, es Entity, y su id es generado mediante la estrategia Identity. Mientras que su única relación es con capítulo, una relación ManyToOne que refleja sobre qué capítulo está aplicado el cargo,