Aitor Juncal Blanco

DOMAIN DRIVEN DESIGN

Clasificación de clases en entity o value object

Usuario : Entity, ya que un usuario se identifica por su nombre de usuario único en el sistema. Además. los usuarios pueden cambiar sus datos (como sus capítulos vistos y sus facturas y estos cambios deben ser monitorizados durante el ciclo de vida de cada usuario)

Serie : Entity, porque cada serie tiene una identidad única dentro del sistema y su ciclo de vida no es estático, sino que debe ser monitorizado (desde su creación hasta que deje de estar disponible). Las series contienen temporadas y capítulos, que pueden ir añadiendose progresivamente.

Temporada : Entity, las temporadas son identificadas mediante un identificador. No puede haber dos temporadas iguales dentro de una misma serie.Por lo que, podría decirse que las temporadas son entidades propias dentro de una serie que hacen de enlace entre los capítulos y una serie, por ello las temporadas tienen operaciones propias y las considero como una entidad que debe ser gestionada y monitorizada a lo largo del tiempo.

Capítulo : Entity, en este caso, tenemos que considerar que cada capítulo va a ser único dentro de nuestro sistema, no puede haber un mismo capítulo en una misma temporada y en una misma serie, como es una entidad única, cada capítulo tendrá su id y será gestionado y monitorizado independientemente del resto de capítulos.

Categoria : Value Object, ya que se trata de un simple enumerado, que no tiene identidad y todas sus posibles instancias están dentro de un conjunto de 3 valores(Silver, Standard, Gold).

Factura : Entity, ya que cada factura es única dentro del sistema y debe ser tratada y gestionada como tal, sabemos que un mismo usuario no puede tener dos facturas en un mismo mes, por lo que debemos tratarlas como entidades independientes.

Cargo : Esta clase podría ser considerada como value Object, ya que simplemente se trata de una colección de entradas que irán en la factura, no son objetos independientes que deben ser monitorizados independientemente, si no que van todos en un pack dentro de la factura por así decirlo.

CapitulosVistos : Entity, es una entidad porque representa el estado del progreso de un usuario con respecto a los capítulos que ha visto. Este estado cambia con el tiempo y necesita ser monitorizado para y es único y respectivo al usuario que pertenece.

Aitor Juncal Blanco

Existencia de aggregates

En la aplicación, se identifican dos aggregates: el aggregate de usuario y el de serie.

Por un lado, en el de usuario, la aggregate root es Usuario, e incluye las clases Usuario, CapitulosVistos, Factura y Cargo. En este agregado, cada usuario contiene una lista de facturas, normalmente una al mes, mientras que cada factura normalmente también tendrá un conjunto de cargos. Mientras que, los capítulos vistos por cada usuario también están estrechamente relacionados con las facturas y los cargos, ya que, realmente cada capítulo que ve un usuario, genera un cargo en la factura del mes actual(siempre que el usuario no sea premium claro)

Por otro lado, el agregado de serie, cuya root es Serie, abarca las clases Serie, Temporada, Capitulo y Categoria. En este caso hay una relación muy clara entre estas clases, ya que una serie tiene un conjunto de temporadas y cada temporada su conjunto de capítulos, mientras que, también tenemos el requisito de que cada serie tenga una categoría asignada.

En este caso, se ha elegido esta separación en dos, ya que aunque el usuario también se relaciona con otras clases del aggregate serie como las propias series(Empezadas, terminadas y pendientes), al final creo que la unión de un usuario con las series, temporadas y capitulos no es tan fuerte como si con sus facturas y cargos. Por otro lado, el aggregate de series se puede ver como algo teóricamente independiente a los usuarios, ya que en la plataforma habrá un conjunto de series(cada una con sus temporadas y capitulos) que podrá luego ser consumida por cualquier usuario.