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.
Serie : Entity, ya que puede identificarse por su nombre. Aunque ante el riesgo de que dos series puedan tener el mismo nombre, he optado por un id numérico único para cada serie
Temporada : Entity, y pueden identificarse a través del número de temporada y a través de la serie correspondiente a la que pertenece.
Capítulo : Entity, identificado por su número y además por la temporada a la que pertenece(que a su vez sabemos a qué serie pertenece)
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 se puede identificar por la fecha y por el usuario a la que va dirigida. Esto es así ya que un usuario nunca podrá tener dos facturas a la vez en un mismo mes.
Cargo : Entity, y en este caso considero que lo más fácil es que cada cargo del sistema tenga un id propio, ya que un mismo usuario puede tener múltiples cargos en una misma factura.

Existencia de aggregates

He identificado dos aggregate en el modelo de dominio : El aggregate de Serie->Temporada->Capitulo, y el de Usuario->Factura->Cargo .

Empezando por el primero, este aggregate tiene como raíz(aggregate-root) a la clase serie. Y a partir de una serie, se puede obtener las temporadas de la misma, y a partir de cada temporada, su lista de capítulos

Por último, en el segundo, el aggregate-root es la clase Usuario, y a partir de ahí, cada usuario tendrá una factura todos los meses, y cada factura tendrá su conjunto de cargos asociados