Practica 1: Domain-Driven Design

Entities y Value Objects

Usuario: Esta clase es claramente una entidad, ya que todos los usuarios de la plataforma deben ser capaces de identificarse de manera univoca. Es imposible que existan en el sistema 2 usuarios exactamente iguales, ya que, si todos los valores coinciden, entonces se trata del mismo usuario. Además, su existencia no está ligada a la presencia de otra clase, sino que de forma independiente puede operar de forma completa.

Usuario normal y Usuario premium: Estas clases también son entidades porque heredan de la clase Usuario.

Factura: Se trata de una entidad porque es un objeto independiente que por si solo tiene cabida en el modelo de dominio. Cada factura que exista en el sistema debe poderse identificar de forma unívoca y se debe poder monitorizar, ya que, a lo largo de su ciclo de vida, irá cambiando. No pueden existir 2 facturas completamente iguales en el sistema ya que se trataría de la misma. Aunque tengan los mismos valores tanto de fecha, como de capítulos vistos o coste, el usuario será diferente. O para 2 facturas del mismo usuario variará la fecha, etc.

Entrada: Yo considero que esta clase es un value object porque su existencia está ligada a la presencia de las facturas. Si no existiera la clase Factura, esta clase no tendría sentido en el modelo de dominio. Además, la clase únicamente representa una colección de datos que ya aparecen en otras clases. No necesitan estar identificadas y diferentes “Entradas” podrían tener sus valores duplicados

Serie: Al igual que ocurre con “Usuario”, “Serie” también es claramente una entidad. Cada serie debe ser identificada de forma unívoca en el sistema y su ciclo de vida es independiente al del resto de clases del dominio, por lo que sería capaz de operar de forma completa sin ninguna dependencia. Además, a lo largo de su ciclo de vida, los objetos de esta clase podrían ir mutando por ejemplo añadiendo más temporadas, y estos cambios deberían ser monitorizados y gestionados.

Estándar, Silver y Gold: También son entidades ya que se trata de clases que heredan de la clase Serie.

Temporada: Esta clase es una entidad porque no es simplemente una colección de datos, sino que se trata de una entidad con un ciclo de vida que debe ser monitorizado y gestionado. Es cierto que puede parecer prescindible esta clase si incluimos el atributo número de temporada en cada capítulo, pero realmente la forma en la que cualquier persona concibe este ámbito es desgranándolo en Serie > Temporada > Capítulo. Además, puede resultar de utilidad esta clase a la hora de realizar diferentes operaciones, ya que, si no existiera, a la hora de hacer búsquedas o modificaciones se deberían recorrer todos los capítulos de una serie.

Capítulo: Esta clase se trata de una entidad porque representa objetos que deben poderse identificar de forma unívoca en el sistema. No pueden existir 2 capítulo iguales, al menos algún atributo será diferente, ya sea la serie, la temporada, el número de capítulo, etc. Si se diera el caso de que existan 2 capítulos con exactamente los mismos campos, deberían unificarse en uno solo. Cada capítulo debe poder ser gestionado de forma independiente.

Creador y Actor: Ambas clases podrían considerarse tanto entities como value objects, pero en mi caso he decidido asumirlas como entities. Esto se debe a que, a pesar de ser clases simples que únicamente contienen el nombre y el apellido, debido a los requerimientos de la plataforma; se trata de entidades completas que deben ser identificadas de forma unívoca, ya que no pueden existir 2 personas que sean completamente iguales, a pesar de tener el mismo nombre y apellido. No se trata de una simple colección de datos que se extrae de otra clase únicamente para mantenerla limpia, porque si fuera necesario se podrían realizar más acciones con esta clase.

FacturacionService: Esta clase es un servicio ya que la tarea de realizar la facturación no pertenece propiamente a ninguna clase del dominio, pero debe ser realizada en el sistema por algún ente.

Aggregates y Aggregate Root

Un aggregate está formado por las clases Usuario, Usuario Normal, Usuario Premium, Factura y Entrada. El aggregate root es la clase Usuario. Todas estas clases están relacionadas y representan a una persona que utilice la plataforma.

Otro aggregate está formado por las clases Serie, Estandar, Silver, Gold, Temporada, Capítulo, Actor y Creador. El aggregate root en este caso es la clase Serie. Si nos fijamos, todas ellas están relacionadas y son depenedientes en algún sentido las unas de las otras.

Realmente, todas las clases están relacionadas, ya que, si no, el modelo de dominio sería disjunto con 2 bloques claramente separados, los usuarios y las series. Pero existen relaciones entre elementos de ambos conjuntos.

Esta división se realiza teniendo en cuenta lo fuertes que son los vínculos entre unas clases y otras. Por ejemplo, si nos ponemos a pensar en otra plataforma como podría ser Steam, si que existen usuarios y estos tienen sus facturas, etc; pero no tienen series, ya que ni siquiera es del mismo dominio.

Un usuario no tiene porque estar relacionado directamente con series, o actores, sino que depende del contexto en el que se utilice.

Del mismo modo, si pensamos por ejemplo en un videoclub, independientemente de si existen o no usuarios, las series tienen sus temporadas, capítulos, etc.