ELEMENTOS DEL DOMINIO

- **Clasificación de los elementos del modelo de dominio en *Entities*, *Value Objects* o *Services***

Entitie: elemento del modelo que tiene entidad por sí mismo independientemente de sus relaciones con el resto de los elementos. Debido a esta identidad intrínseca:

- Vamos a querer identificar los objetos de este tipo de elementos de alguna forma
- Duplicidades de objetos de este elemento causarían un problema grave.
- Los objetos sufrirán cambios durante su proceso de vida

Value Objects: elementos del modelo que simplemente representan valores. Existen como clases porque el lenguaje utilizado no permite otra forma de definirlos.

Service: elementos del modelo utilizados para recoger operaciones que tienen que ver con el dominio pero que no encajan de manera natural dentro de ningún otro elemento.

Clase	**Tipo de elemento**	**Explicación**
Usuario	Entitie	Los usuarios son una pieza fundamental del dominio. Cada usuario debe poder distinguirse unívocamente del resto de usuarios de la aplicación porque debemos tener las garantías de que las consecuencias de cualquier acción que realice un usuario dentro de la plataforma le sean asignadas al usuario que efectivamente ha realizado dichas acciones. Tener usuarios duplicados en el sistema, por lo tanto, puede generar problemas que alteren el correcto funcionamiento de la aplicación en caso de que, por ejemplo, se le modifique su tipo de tarifa. Si se modifica esta característica, debemos tener las garantías de que todos los capítulos se le cobren de acuerdo a su tipo de usuario. Además, los usuarios tendrán un ciclo de vida muy largo dentro del sistema y cuyo estado va a querer ser monitorizado ya que puede verse modificado porque el valor de sus atributos puede cambiar, como ocurre con la característica del tipo de usuario.
Serie	Entitie	De la misma manera que con los usuarios, las series son una parte fundamental del dominio y deben ser identificables de manera unívoca. Cada serie tiene asociado un precio y un conjunto de capítulos, por lo que es clave que cuando se vaya a ver una serie, se acceda a los capítulos que efectivamente pertenecen a esa serie en concreto y que se cobre por la visualización de cada capítulo el precio asignado al tipo de serie que sea.

		Además, tener objetos de series duplicados en el sistema puede causar que determinados atributos de una serie que se modifiquen no sean percibidos de la misma manera por todos los usuarios que estén visualizando dicha serie. Si se modifica el precio de los capítulos de una serie, la modificación tiene que ser vista por todos los usuarios que la estén viendo, por lo que no pueden existir varios objetos de la serie en el sistema. Los usuarios que estén viendo la misma serie deben compartir el objeto de esa serie.
Temporada	Value Object	La temporada a la que pertenece cada capítulo es una información que no va a ser modificada, ya que nunca va a variar la temporada a la que pertenezca cada capítulo, y que representa los valores de la temporada a la que pertenece cada capítulo.
Capítulo	Entitie	Los capítulos van a tener que ser identificados de manera unívoca porque van a tener un ciclo de vida muy largo en el que será monitorizado, ya que puede cambiar el valor de algunos de sus atributos. No poder identificarlos puede provocar errores de funcionalidad como presentar los capítulos a los usuarios de forma desordenada, no guardar correctamente el último capítulo visto de una serie o marcar una serie como terminada cuando aún no se ha visto su último capítulo. Además, tener duplicidades de objetos de capítulos en el sistema también podría afectar a la funcionalidad en caso de que se tenga que modificar, por ejemplo, su enlace de visualización. Esto podría producir que unos usuarios que tengan un capítulo disponible para su visualización y otros no.
TrabajadorSerie	Value Object	Las personas que trabajan en una serie (creadores y actores principales) no suponen un elemento importante dentro de la aplicación, sino que se presentan como información adicional. No son elementos que vayan a tener un ciclo de vida durante el que vayan a sufrir modificaciones, sino que simplemente representa un valor para un atributo de la clase Serie que el lenguaje utilizado no permite expresar o representar de otra forma. En el hipotético caso de que los creadores o actores principales de una serie se mostrasen erróneamente, el impacto sobre el funcionamiento de la aplicación sería bastante bajo.
VisualizacionCapitulo	Value Object	Los objetos que representan las visualizaciones que los usuarios hacen de los capítulos de las series van a utilizarse para registrar estas visualizaciones y no van a tener una vida en la que vayan a sufrir modificaciones ni vaya a haber que monitorizarlos. Dos usuarios que vean un capítulo van a generar ambos el mismo objeto del tipo VisualizacionCapitulo, que podrían incluso intercambiar entre ellos. Además, no tendría sentido que hubiese solamente un objeto de este tipo compartido por todos los usuarios de la aplicación, sino que cada usuario tendrá los

		objetos que él mismo genere que servirá como información adicional del propio usuario.
VisualizaciónSerie	Value Object	De la misma manera que en el caso anterior, los objetos de la clase VisualizacionSerie sirven para registrar el último capítulo visto de cada serie por parte de cada usuario, por lo que cada usuario tendrá su propio objeto del tipo VisualizaciónSerie que podría incluso intercambiar con otro usuario. Además, al igual que antes, estos objetos se utilizan como una manera de almacenar información adicional en relación a cada usuario.
Factura	Entitie	En el caso de las facturas, no se puede permitir tener varios objetos de una factura porque podría generar problemas de consistencia a la hora de añadir entradas a la factura y provocar consultas erróneas de usuarios sobre sus facturas. Además, para que la aplicación funcione como se espera se necesita que las facturas sean identificables de manera unívoca en todo el sistema y que exista la capacidad de poder distinguir unas facturas de otras. En caso de no poder identificar unívocamente cada factura existente en la aplicación, no habría garantías de realizar los cobros las facturas correctamente: se podrían asignar facturas a usuarios de forma errónea y cobrarles un importe que no les corresponde.
Categoría	Entitie	Todas las series de la misma categoría deben compartir entre ellas el mismo objeto Categoría, de manera que si en algún momento se modificase el precio de los capítulos relativos a las series pertenecientes a dicha categoría, tuviésemos las garantías de que este cambio estuviese afectando a todas estas series.
CuentaBancaria	Value Object	Al igual que ocurre con la clase TrabajadorSerie, la clase CuentaBancaria representa un valor para un atributo de la clase Usuario que el lenguaje utilizado no permite expresar o representar de otra manera.
FacturasHelper	Service	La clase FacturasHelper es un elemento utilizado para encapsular la operación de realizar los cobros a los usuarios, ya que dicha tarea no encajaba de manera natural en ninguna otra clase del modelo de dominio. No encaja bien en la clase Usuario, porque no tiene sentido que un usuario realiza la acción de cobrarse su factura a sí mismo, y tampoco encaja en la clase Factura, porque esta clase encapsula características y acciones relativas a una factura, por lo que no tendría sentido meter aquí una operación que fuese el cobro de todas las facturas de los usuarios de la aplicación.

- **Identificación de los *aggregates* del modelo creado, destacando su *aggregate root*.**

Aggregate: conjunto de *entities* y *value objects* con una frontera clara con el resto del modelo de dominio y una colección de invariantes bien definidos que deben preservarse dentro de dicho conjunto.

Agreggate root: *entity* que controla el ciclo de vida del resto de los elementos del *agreggate* y que contiene la información necesaria para encargarse de la preservación de los invariantes.

Agreggate	**Agreggate root**	**Explicación**
Usuario, CuentaBancaria, Factura, VisualizacionSerie, VisualizacionCapitulo	Usuario	Todas las clases de este agreggate tienen sentido siempre que exista al menos un usuario al que estén asociados. No tendría sentido que existiesen facturas que no estuviesen asociadas a un usuario, ya que es el usuario con sus acciones el que provoca la generación de facturas. De la misma manera, los objetos para las visualizaciones deben existir asociadas a un usuario que sea el que vea los capítulos de las series. Sin usuarios que vean capítulos, no existirían objetos de esas dos clases. Por tanto, estas cuatro clases forman un bloque dentro del modelo de dominio que gira en torno a la clase Usuario. Tampoco tiene ningún sentido que existan cuentas bancarias en el sistema que no estén asociadas a ningún usuario. El agregate root es la clase Usuario porque del usuario dependen el resto de elementos. Sin usuarios, no existen facturas, visualizaciones ni cuentas bancarias, y en el momento en el que se borre un usuario, probablemente no tenga sentido seguir almacenando ninguno de esos tres elementos. Por tanto, el ciclo de vida de un usuario condiciona el ciclo de vida en el sistema de sus facturas, visualizaciones y su cuenta del banco.
Serie, Temporada, Capitulo, TrabajadorSerie, Categoria	Serie	De igual forma, las temporadas, los capítulos, las categorías y los actores están asociados a las series, y si no existiesen las series, no existirían ninguno de los elementos anteriores. Una temporada y un capítulo, lógicamente, siempre van a estar asociados a la serie a la que pertenezcan, y necesitan obligatoriamente que esa serie exista para que puedan existir ellos también. Por otro lado, los trabajadores de una serie necesitan que esa serie exista para poder trabajar en ella, y para que existan categorías tienen que existir series que pertenezcan a esas categorías. Por lo tanto, estas 5 clases forman otro bloque dentro del modelo de dominio que gira en torno a la clase Serie. El agreggate root es la clase Serie porque es el elemento que condiciona el ciclo de vida del bloque. Sin series no existirían el resto de elementos, ya que no tiene ningún sentido tener temporadas y capítulos que no pertenezcan a ninguna serie, trabajadores de una serie que no existe o categorías sin series.